VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Jaroslav Bošeľa

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH A IMPLEMENTACE SÍŤOVÉHO KOLEKTORU
DESIGN AND IMPLEMENTATION OF NETWORK COLLECTOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jaroslav Bošeľa**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Václav Oujezský, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Jaroslav Bošeľa ***ID:*** 173617
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Návrh a implementace síťového kolektoru

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte a v teoretické části diplomové práce popište typy informačních zpráv síťových zařízení, zejména protokolů NetFlow a IPFIX (Internet Protocol Flow Information Export). V praktické části diplomové práce navrhněte a implementujte příjem a uložení NetFlow verze 9 v jazyku Python verze 3 a vyšší. Toto téma vyžaduje hlubší znalosti z oblasti sítí, zejména zařízení Cisco a znalost jazyka Python. Pro vlastní vypracování práce je k dispozici vývojový server, reálná cisco zařízení a simulační prostředí včetně GNS3, případně EVE-NG.

DOPORUČENÁ LITERATURA:

[1] Introduction to Cisco IOS NetFlow - A Technical Overview. CISCO SYSTEMS, INC. CISCO [online]. 2012 [cit. 2019-09-07]. Dostupné z: https://bit.ly/1NaU6lT

[2] PILGRIM, Mark. Ponořme se do Python(u) 3: Dive into Python 3. Praha: CZ.NIC, c2010. CZ.NIC. ISBN 978-80-904248-2-1.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Václav Oujezský, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Táto diplomová práca sa zaoberá popisom informačných protokolov sieťového toku, najmä popisom protokolu Cisco NetFlow verzie 9. Popisuje jeho vlastnosti, formát správ a atribúty prenášaných dát. Práca sa zameriava hlavne na prenášanú šablónu NetFlow v9, ktorá definuje polia a dáta v následnom dátovom toku. Podstata práce spočíva v implementácií jednoduchého NetFlow v9 parseru v programovacom jazyku Python, jeho testy na zachytené UDP dáta zo súboru a testovanie na porte vývojového serveru v laboratóriu. Dáta je v rámci implementácie možné ukladať aj do pripravenej databázy, ktorá je definovaná ako možný výstup z parsovania a zachytávania.

KĽÚČOVÉ SLOVÁ

Sieťová prevádzka, prenosy, protokoly, dátový tok, analýza, Netflow, Cisco, Netflow v9, pakety, IPFIX, sFlow, NetStream, OpenFlow, CFLOW, Python, polia, export, kolektor

ABSTRACT

This master's thesis deals with description of information protocol of network flow, mainly definition of Cisco NetFlow version 9. Describes it's features, message format and attributes of transmitted data. The thesis is primarly focused onto NetFlow v9 transmitted template, which defines fileds and data in consecutive data flow. The essence of the thesis consists in implementation of simple NetFlow v9 parser, which has been programmed in Python prog.language, it's tests of captured UDP data from file and port capture testing on development server in lab. There is a possibility of saving captured and parsed data into prepared database within implementation as output from capturing.

KEYWORDS

Network traffic, transmission, protocols, data flow, analysis, NetFlow, Cisco, NetFlow v9, packets, IPFIX, sFlow, NetStream, OpenFlow, CFLOW, Python, fields, export, collector

BOŠEĽA, Jaroslav. *Návrh a implementace síťového kolektoru*. Brno, 2020, 69 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Václav Oujezský, PhD.

VYHLÁSENIE

Vyhlasujem, že svoju diplomovú prácu na tému „Návrh a implementace síťového kolektoru“ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som sa poďakoval vedúcemu diplomovej práce pánovi Ing. Václavovi Oujezskému, Ph.D. za veľkú pomoc, odborné vedenie, rýchle konzultace v každom čase, nesmiernú trpezlivosť a podnetné návrhy k práci. Zároveň sa mu chcem poďakovať za vloženú dôveru, poskytnutie priestoru a prostriedkov na testovanie v školskom laboratóriu.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V dnešnom svete rastúcej internetovej a sieťovej prevádzky sa kladú dôrazy na analyzovanie a monitorovanie sieťového toku. Na začiatku vzniku samotného Internetu, sa analýzy využívali hlavne na lepšie smerovanie, zvyšovanie rýchlosti a zlepšovanie samotného prenosu sieťových informácií - dát. S postupným veľkým nárastom používateľov, spojovacích bodov a enormným nárastom prevádzky, sa analýzy a monitorovanie prevádzky začalo využívať aj na odhaľovanie bezpečnostných rizík a slabých miest v sietí. Pomocou protokolov dokážeme odhaľovať zraniteľné sieťové uzly, ich abnormálne správanie, prípadne netypický priebeh toku. Exportné sieťové protokoly nám teda umožňujú odhaľovanie vnútorných a vonkajších incidentov, zdokonaľovanie samotného prenosu a zabezpečenie vyrovnaného zaťaženia siete.

Táto diplomová práca sa zameriava sa na popis internetových otvorených protokolov IPFIX a najmä NetFlow. Tieto protokoly sa využívajú hlavne na export informácií a atribútov o sieťovom toku, na optimalizovanie prevádzky, charakterizovanie sieťových operácií a v neposlednom rade na nastavenie služieb pre dané spojenie. V reálnom čase dokážu odhaliť slabšie miesta v sietí, dominantné zdroje prevádzky, zaistiť lepšie nastavenie siete a poskytnúť tak žiadúce štatistiky pre sieťových administrátorov ako nevyhnutný viditeľný sieťový nástroj.

V práci implementujem protokol NetFlow verzie 9, čo je posledná inovácia protokolu NetFlow od firmy Cisco. NetFlow verzie 9 prináša rozdielny typ informačných správ oproti verziám 1 a 5. V práci popisujem rozdiel medzi jednotlivými verziami, ich atribúty a výhody. Popíšem architektúru protokolu NetFlow, zloženie jednotlivých správ v daných verziách protokolu. Tento protokol je proprietárny protokol firmy Cisco, preto sa na testovanie budú využívať reálne Cisco zariadenia, vývojový server a simulačné prostredie v školskom laboratóriu.

Programovanie a zachytávanie exportov protokolu prebiehalo v jazyku Python od verzie 3 a vyššej, v programe PyCharm. Zameranie diplomovej práce je hlavne na export jednotlivých správ, konkrétnych šablón, informácií a dát ktoré plnia jednotlivé polia definované v šablónach.

V rámci cieľov diplmovej práce bol naprogramovaný a implementovaný netflow súbor v jazyku Python 3.7, ktorý plní funkciu kolektora a parsera pre zachytávané dáta z exportéra. Súbor sa spúšťa v príkazovom riadku a je schopný zapísať a uložiť dáta do jednoduchej naprogramovanej databázy.

1 Teoretická časť práce

1.1 Sieťová prevádzka

Sieťová prevádzka je termín, ktorý popisuje prechod množstva dát sieťou v určitý čas. Tieto dáta sú v našom prípade Internetu, teda najmä paketových sietí, prezentované práve paketmi. Paket je jednotka dát, v ktorej sú zabalené dáta a informácie z protokolov vyšších vrstiev. Samotná sieťová prevádzka teda nepopisuje správanie jednotlivých konkrétnych tokov, popisuje skôr správanie na danej sietí ako také, teda súbor komunikácií jednotlivých tokov, protokolov, portov a služieb.

Sieťová prevádzka je teda komunikácia medzi sieťovými bodmi, ktorých môže byť ako viac na strane zdroja, tak môže existovať viacero cieľov, danej sieťovej komunikácie. Rovnako môže existovať sieťová prevádzka medzi viacerými sieťami, dokonca to nemusia byť nutné len siete paketové, ale komunikácia môže prechádzať medzi ATM sieťami, prípadne spojovanými okruhmi a inými. Samozrejmosťou je zabalenie prechádzajúcich dát do formátu daného typu komunikácie. V práci sa však zameriavam čisto na paketovú prevádzku v internetových sieťach, v našom prípade simulované virtuálne spojenie vrámci laboratória.

Pojem sieťová prevádzka sa používa najmä v prípade definície vlastnosti komunikácie na sledovanej časti siete, medzi uzlami, prípade pre celú sieť. Sieťová prevádzka je označenie pre sériu dátových tokov v danom čase. Sieťová prevádzka je hlavný komponent pre optimalizovanie jednotlivých sietí, ich analýzu, kontrolu a simuláciu ich správania. Samotná sieťová prevádzka teda vytvára veľmi žiadúce atribúty pre sieťových administrátorov a pre organizácie spravujúce siete. Exportované informácie o sieťovej prevádzke dávajú do rúk odborníkov kľúčové nástroje pre nastavovanie priepustnosti siete, počtu komunikačných bodov, šírky priradeného pásma, prípadne odozvy. Množstvo sieťovej prevádzky a jej vlastnosti sú určujúcimi parametrami pre implementáciu rôznych sieťových topológii, ktoré sú na tieto parametre citlivé. [1]

Sieťovú prevádzku môžeme zoširoka klasifikovať do nasledujúcich kategórií:

- **Real-time sieťová prevádzka** - Real-time znamená prevádzku v danom čase prenosu, táto prevádzka vyžaduje nízke oneskorenie paketových dát idúcich za sebou, preto sa pre túto prevádzku rezervuje isté pásmo kde je zabezpečené, že dáta budú nasledovať rýchlo po sebe a nedôjde k veľkej strate paketových dát. Veľkosť dát sa zadefinuje na začiatku prenosu pre danú kvalitu prenosu a počas prenosu sa to nemení pokiaľ to nevyžaduje spojenie, zároveň sú to dáta s malým oneskorením(delay), ktoré spolu s istou tolerovanou stratou malej časti dátových paketov, ktorá ešte významne neovplyvňuje kvalitu a príjemca je v určitej miere schopný kompenzovať pomocou opravných mechanizmov.

Pri tomto type prevádzky však nie je možné pakety posielať rôznymi poradiami v rôznom čase, pretože dáta sú vykresľované de facto ihneď v danom čase prenosu-streamu, dáta ktoré prídu po čase tohto prenosu alebo spojenia sú pre komunikujúce strany zbytočné. Alokované pásmo je teda rezervované rezervačnými protokolmi na dobu prevádzky medzi príjemcom a vysielajúcim. Príkladom takejto sieťovej prevádzky môže byť VoIP, prípade videokonferencia.

- **Non-Real time prevádzka** - Normálne definovaná prevádzka, kde paketové dáta prechádzajú od zdroju k cieľu rôznymi cestami a sieťami. Na rozdiel od prevádzky real-time, kde dochádza k rezervácií pásma v danom okamihu komunikácie, a je vyžadované čo najmenšie oneskorenie dát plus pre danú kvalitu vyžadovaná pevná veľkosť paketov, pri normálne definovanej prevádzke nedôjde k rezervácií pásma ale využitie voľného pásma v daný okamih. Veľkosť takéhoto pásma sa môže meniť a teda dáta budú sieťou putovať v rôznom čase, rôznych veľkostiach a rôznymi cestami. Podľa druhu smerovacích protokolov a mechanizmov sa bude prenos prispôsobovať podmienkam na sietí, veľkosti pásma a jeho priepustnosti. Zároveň je na prenosových mechanizmoch a vyjednaných protokoloch oboch komunikujúcich strán aby sa snažili doručiť všetky dáta a prispôsobili sa danej situácií v čase prenosu. Obidve strany disponujú mechanizmami na potvrdzovanie a kontrolu doručenia dát - ich celistvosti a kompletnosti, v prípade straty dát aj ich opakovaný prenos. Príkladom takejto prevádzky môže byť HTTP, prenos súborov FTP alebo e-mailové služby elektronickej pošty. [2]

1.2 Typy prenosov

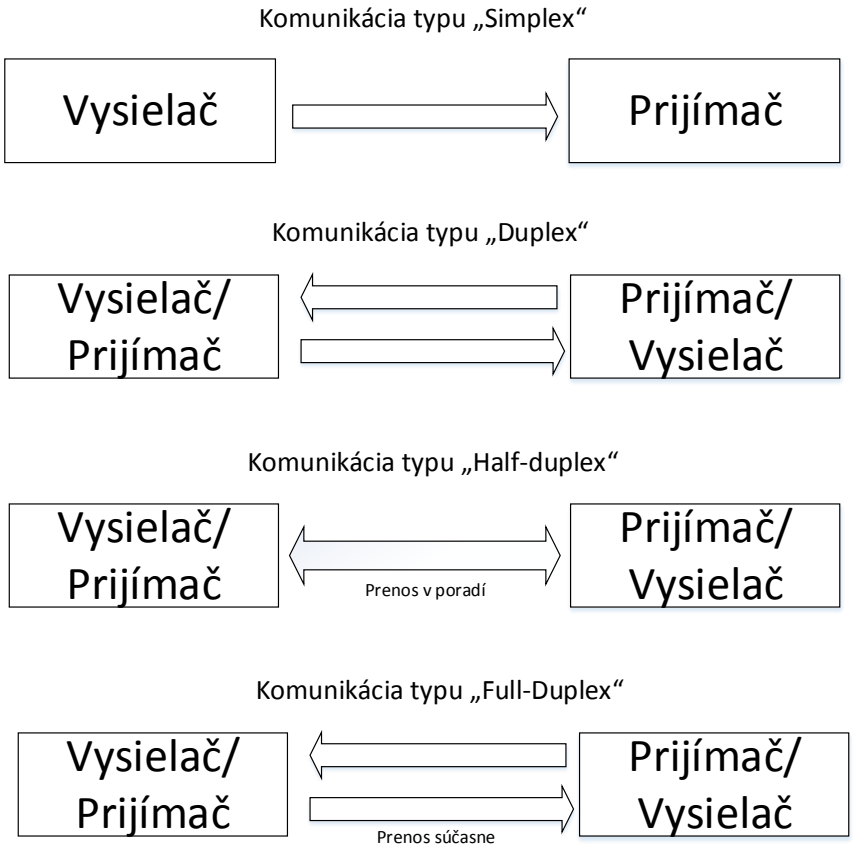
Všeobecne sú zbernice a siete sú navrhnuté na komunikáciu medzi jednotlivými prepojenými bodmi. Takáto komunikácia nesie informácie, prípadne dáta ktoré sú prenášané dátovým tokom. Na najzákladnejšej úrovni rozlišujeme prenos podľa schopnosti komunikácie prepojených bodov. [3]
Teda prenosová komunikácia nielen v sieťovej prevádzke ale všeobecne sa delí na dve základné úrovne:

- **Simplex prenos** - jedná sa o jednoduchú komunikáciu od nastaveného kanála vysielača ku nastavenému kanálu prijímača. Komunikácia je jednosmerná a komunikačné body nemenia svoje role počas komunikácie. Príkladom takejto komunikácie je napr. TV vysielanie, príjem rádiového vysielania. V moderných sieťach sa napríklad používa ako jednosmerný kanál po optickom spojení.
- **Duplex prenos** - komunikačné body môžu komunikovať v oboch smeroch, teda dáta aj prijímať aj posielať.

Duplex prenos sa delí ešte na dve ďalšie kategórie:

- **Half Duplex** - tzv. polovičný duplex. V danom čase môže dáta posielať, prípadne prijímať len jedna strana. Komunikačný kanál je teda obsadený jednou stranou a druhá strana ak chce posielať tak musí čakať v poradí na skončenie vysielanej komunikácie od prvej strany.
- **Full Duplex** - tzv. plný duplex. komunikácia medzi bodmi je obojsmerná a zároveň je možné prijímať aj posielať súčasne. Komunikačným kanálom sú teda prenášané dáta obidvoch bodov v obidvoch smeroch naraz. Full-Duplex kanály môžu byť súčasťou simplex liniek, prípade používať jeden kanál pre obojsmernú prevádzku. Spojenie full duplex je vždy medzi dvoma bodmi, ak máme viac bodov tak potrebujeme odpovedajúci počet spojení medzi týmito bodmi. [4]

Základné typy prenosov môžeme vidieť na obrázku 1.1

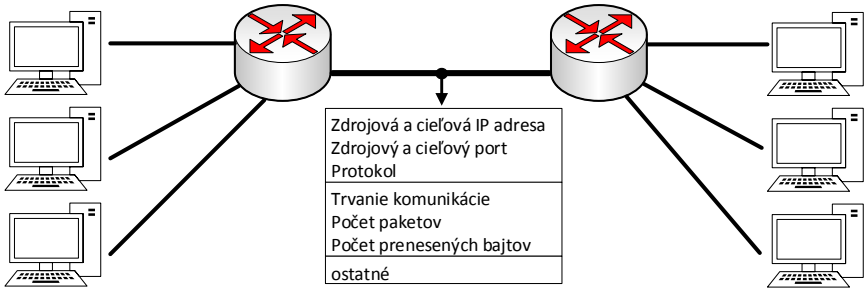


Obr. 1.1: Všeobecné typy komunikačného prenosu

1.3 Sieťový dátový tok

Pri pojme sieťový dátový tok, hovoríme o súbore dát ktoré smerujú medzi komunikačnými bodmi s konkrétnymi atribútmi v daný konkrétny čas. V prípade dátového toku, hovoríme o prenose paketov, ktoré zdieľajú 5 obvykle spoločných prvkov - cieľovú a zdrojovú IP adresu, cieľový a zdrojový port a IP protokol. Tieto pakety však môžu v sebe niesť aj iné informácie - napríklad dobu komunikácie, typy služieb, veľkosť paketov, počet prenesených bajtov, kvalitu a prioritu služieb prípade iné informácie. Príklad identifikácie dátového toku môžeme vidieť na obrázku 1.2.

Pojem dátový tok sa používa aj v prípade agregácie viacerých tokov, avšak musíme mať na pamäti, že tok sa vyznačuje tým, že dáta v ňom idú v jednom smere. Odlišnosť od relácie je v tom, že v relácia môže obsahovať viacero viacsmerných tokov. Napríklad, ak sa klientský počítač pripojí na web server a snaží sa stiahnúť súbor, tak sa vytvorí HTTP(prípadne HTTPS) relácia, ktorá však v sebe nesie dva dátové toky - jeden od klienta k serveru a jeden od servera ku klientovi. Každý tok takto odráža prenos toho druhého toku, zrkadlia sa navzájom. [5]



Obr. 1.2: Dátový tok a jeho vlastnosti

1.3.1 Využitie záznamov o dátovom toku

Záznamy z dátového toku sú veľmi užitočný nástroj na to, aby človek ktorý vykonáva manažment a dohľad nad danou sieťou mal prehľad o prenosoch dát, výkonnosťou siete a jej slabých miestach. Rovnako dnes implementujeme automatizované systémy, ktoré dokážu na základe dátového toku vyhodnocovať pripojenie, prípadne využitie sieťových zdrojov pre daný tok. Informačné protokoly, ktoré dokážu analyzovať dátový tok sú dnes využívané na analýzu veľkého množstva metadát. Zároveň tým získavame informácie o veľkosti tokov, ich trvanie, časové úseky jednotlivých datagramov rozličných protokolov.

Pomocou záznamov o toku dokážeme analyzovať využitie šírky pásma, jeho zahltenie, prípadne narábanie so sieťovými prostriedkami, rovnako dokážeme sledovať vplyv jednotlivých zmien v sietí na dané toky a aplikácie. Jednotlivé informačné protokoly dátového toku sú schopné užitočne vykresľovať chovanie jednotlivých aplikácií, takto je možné sledovať napríklad bezpečnosť jednotlivých tokov ale aj bezpečnostnú stránku celej siete pred útokmi alebo zlyhaniami. Ak by došlo k abnormálnemu chovaniu niektorých aplikácií a začali by zahlcovať kanál, tak pomocou záznamov o toku by bolo možné podobnému správaniu predísť, prípadne na základe informácií o toku, rozbehnúť vyšetrovanie zlyhania alebo útoku, nakoľko by sme mali k dispozícií informácie kde k zlyhaniu došlo, aký typ aplikácií, aký protokol a typ datagramov sa využíval, čo bol zdroj, cieľ a trvanie takéhoto toku. Informačné záznamy o toku nám pomáhajú aj v rozvoji siete, vieme sa dozvedieť kde je vhodné implementovať jednotlivé aplikácie a služby, rovnako nám pomôže identifikovať sieťovú produktivitu a využitie prostriedkov pre sieťové služby a ich následne zlepšenie do budúcnosti, prípadne implementovanie ďalších služieb. [6]

Informácie a záznamy o toku v neposlednom rade umožňujú nastavenie služieb pre daného zákazníka a ich prípadné účtovanie podobne ako pri telefónnych hovoroch. Príkladom dnes môžu byť dátové paušály, ktoré umožňujú využívanie niektorých aplikácií a služieb, ktoré na základe daného toku využívajú iné dátové balíčky a umožňujú odlišné účtovanie služieb a prispôsobenie ponuky.

1.4 Informačné protokoly dátového toku a ich informačné správy

V dnešnej dobe mnoho výrobcov používa svoje proprietárne riešenia informačných správ o toku a rovnako aj proprietárne sieťové architektúry založené na týchto protokoloch. Informačné protokoly sa rovnako líšia aj základňou informácií ktorými disponujú a formátmi jednotlivých správ. Medzi najznámejšie patrí protokol:

- *Netflow* - Proprietárny protokol vyvinutý spoločnosťou Cisco a jedna z hlavných tém tejto práce. Netflow spracuváva toky primárne pre IP protokol, s tokmi nižších vrstiev nevie dokonale pracovať. Protokol je používaný ďalšiou radou výrobcov(Juniper, Huawei, Alcatel Lucent, Nortel,Enterasys) práve na zbieranie dát o toku a ich analýzu. Protokol nebol v zásade definovaný v RFC, avšak jeho posledná verzia č.9 bola štandardizovaná v RFC 3954.[7] Posledná verzia v9 je aj značne škálovateľná pomocou template(šablón), a umožňuje detailne si prispôsobiť exporty o tokoch. Podrobne je protokol opísaný v sekcií 1.5

- *IPFIX* - IP Flow Information Export. Teda sa jedná o exportový protokol IP toku, ktorý je definovaný v RFC 5101, neskôr rozšírený v RFC 7011 [14]. Nezávislý štandard, je to praktický open source riešenie vychádzajúce z protokolu NetFlow. Detailne je definovaný v sekcií
- *sFlow* - Sample flow protokol - vzorkovací protokol s minimálnym dopadom na výkonnosť systému. Protokol môže vzorkovať aj iné ako IP toky na tretej vrstve. Vzorkuje v zásade pakety, neanalyzuje celý tok. Rovnako je podporovaný radou výrobcov. Definovaný v RFC 3176 [8]
- *OpenFlow* - Programovateľný sieťový protokol vyvinutý špeciálne pre SDN (Software Defined Networks – Softvérovo definované siete).
- *NetStream* - Je to ekvivalent protokolu NetFlow avšak od výrobcu Huawei. Založený je na veľmi podobnom prinicípe ako NetFlow štandard. Líši sa najmä v pomenovaní komponentov pre zachytávanie a analýzu toku.

1.5 Netflow Protokol

Netflow protokol je funkcia spoločnosti Cisco predstavená pôvodne pre ich proprietárne zariadenia okolo roku 1996. Je to vstavanou súčasťou Cisco IOS softvéru na charakterizovanie sieťovej prevádzky. Slúži na zachytávanie a zasielanie štatistík o paketoch, ktoré sú smerované sieťovými zariadeniami v dátových tokoch. NetFlow identifikuje dátové toky pre odchádzajúce aj prichádzajúce IP pakety, avšak nevytvára a nenastavuje žiadny protokol. Nevyžaduje žiadne vonkajšie zmeny, ani v paketoch a ani v sieťových zariadeniach. Je kompletne transparentný pre už existujúcu sieť, pre koncové stanice, aplikačný softvér aj pre sieťové zariadenia ako LAN prepínače(Local Area Network – Lokálna sieť) a smerovače. NetFlow primárne vznikol ako technológia pre účtovanie služieb a bezpečnostná sieťová technológia, avšak značne sa rozšíril do podoby exportného a informačného protokolu o dátovom toku.[9] Benefity protokolu NetFlow spočívajú hlavne v bohatosti štatistík o toku, ktorý zachytáva a ich využití.

Sieťové aplikácie a monitorovanie užívateľov - Údaje z protokolu NetFlow umožňujú zobraziť podrobné informácie o časovom a aplikačnom využití siete. NetFlow poskytuje informácie, ktoré pomáhajú v plánovaní a alokovaní sieťových a aplikačných zdrojov, týmto nám umožňuje monitorovať sieť v takmer reálnom čase a predchádzať tak potencionálnym bezpečnostným hrozbám a porušeniam pravidiel v sieti. Používa sa na zobrazenie prenosových vzorcov a zobrazení na základe aplikácií. NetFlow umožňuje proaktívne zistiť problém v sietí, zacieliť naň a efektívne a rýchlo ho vyriešiť.

Sieťové plánovanie - NetFlow záznamy umožňujú zachytávať dátové toky v dlhom časovom horizonte, čo otvára možnosti sledovať a predvídať rast prevádzky na sietí, prípadne plánovať vylepšenia a odstraňovať slabé miesta siete. Údaje používame aj na plánovanú optimalizáciu, ktorá zahŕňa peering(prepojenie dvoch administratívne oddelených komunikačných bodov), vylepšenie chrbticovej siete(Backbone network – Nosná chbrticová sieť) a hlavne na optimalizáciu politiky smerovania. S využitím záznamov z NetFlow sme schopní minimalizovať náklady pri maximalizovaní výkonu, spoľahlivosti a kapacity siete. NetFlow je schopný detegovať nechcenú WAN prevádzku(Wide Area Network – Rozľahlá oblastná sieť), overiť šírku pásma a QoS(Quality of Service – Kvalita služby) a analyzovať nové sieťové aplikácie. Poskytuje cenné informácie ktoré znižujú náklady na spravovanie siete.

Útok odmietnutie služby a Analýza Bezpečnosti - Pomocou dát z NetFlow protokolu sme schopní identifikovať a klasifikovať DoS útoky(Denial of Service – Odmietnutie Služby), vírusy a počítačových červov v reálnom čase, pretože dokážeme sledovať anomálie v sieťovom správaní a výkone, ktoré nám reflektujú dáta z NetFlow exportov. Pomocou týchto exportov a ich dát, môžeme získavať cenné informácie na vyšetrenie, zoznámenie sa a pochopenie bezpečnostných incidentov a zásahov, ktoré sa odohrali v minulosti. To nám umožňuje predchádzať útokom a incidentom v budúcnosti.

Účtovanie a fakturácia - Z údajov NetFlow môžeme detailne a podrobne merať využívanie zdrojov pre flexibilné účtovanie. Údaje o toku zahŕňajú podrobnosti ako IP adresy, počítadlo paketov a prenesených bajtov, časové značky, typy služieb a aplikačné porty pre aplikácie. Poskytovatelia služieb môžu tieto informácie použiť pre fakturáciu služieb na základe doby pripojenia, využitej šírky pásma, využívaní aplikácií alebo kvality služieb. Podnikoví zákazníci môžu tieto informácie využívať na rozdelenie nákladov pre oddelenia alebo podľa využívania zdrojov.

Využitie exportov o sieťovej prevádzke - NetFlow poskytuje informácie o sieťovej prevádzke vo vnútri autonomného systému, takýto zachytený prenos môže byť využitý na pochopenie trendov v prevádzke medzi zdrojom a cieľom a ich cielením vyváženie záťaže naprieč alternatívnymi trasami, prípadne na smerovanie po preferovanej trase. Pomcou exportov, máme prehľad o objeme prenosu medzi jednotlivými peer komunikačnými bodmi alebo tranzitnými bodmi a môžeme tak vyhodnotiť či sú nastavené služby spravodlivé.

Ukladanie a extrahovanie NetFlow dát - Dáta z NetFlow môžu byť uložené a neskôr analyzované pre marketingové a zákaznícke účely. Pomocou týchto dát môžu poskytovatelia zistiť aké aplikácie a služby používajú interní alebo externí zákazníci a zacieliť na týchto zákazníkov. Zároveň umožňujú poskytovateľom zlepšiť služby a aplikácie ktoré sú u zákazníkov populárne a vytvárať na ne reklamu.

Dáta z NetFlow sú veľmi žiadané pre podniky a poskytovateľov z dôvodu toho, že umožňujú zistiť - kto, čo, kde a ako dlho využíval vo svojom dátovom toku. [9] Z marketingového hľadiska sú to neoceniteľné informácie, majúce veľkú hodnotu pre analytické firmy.

1.5.1 História protokolu NetFlow

Protokol NetFlow od svojho vzniku prekonal viacero verzií, niektoré verzie boli len pracovné a neboli nikdy vydané, podrobnejší prehľad je v tabuľke.[10]

Tab. 1.1: Verzie NetFlow Protokolu

Verzia	**Popis**
v1	Prvá, dnes už prekonaná a zastaralá verzia, podporovala len IPv4 bez IP masky a čísiel AS(autonómnych systémov)
v2 - v4	Interné pracovné verzie spoločnosti Cisco, nikdy nevydané
v5	Najpoužívanejšia a najrozšírenejšia verzia, implementovaná aj radou iných výrobcov podporuje však len IPv4 toky
v6 - v8	Vo verzí 6 boli pridané informácie o zapuzdrení, verzia 7 priniesla zdrojové informácie o smerovačoch a vo verzií 8 bola pridaná agregácia tokov.
v9	Najnovšia verzia založená na šablónach(template), pridaná podpora pre IPv6 pre MPLS. Obsahuje aj ďalšie položky. Možnosť modifikácie záznamov

1.5.2 Architektúra NetFlow

NetFlow architektúra je založená na obvykle dvoch, prípadne troch komponentoch. Je to *NetFlow Exportér*, *NetFlow kolektor* a prípadne *Management Console*(konzola pre manažment).

Exportér

Zariadenie pracujúce s protokolom NetFlow. Exportér monitoruje pakety ktoré prechádzajú sledovaným bodom a vytvára z nich záznamy o tokoch. Z týchto tokov generuje NetFlow dáta, ktoré sú exportované do kolektorov toku. NetFlow exportéri sa delia na tri základné kategórie:

- **Smerovače, prepínače a firewally** - Sieťové zariadenia klasickej infraštruktúry, ktoré sú schopné zachytávať a exportovať NetFlow dáta do kolektorov. Výhodou týchto zariadení je ich centrálna pozícia v sietí, teda dokážu zachytávať množstvo dát, ktoré cez tieto zariadenia prechádza. Vo veľa prípadoch sú zariadenia vyšších rád výrobcov už pripravené na generovanie NetFlow záznamov.
- **Dedikované senzory a sondy** - Rôzni výrobcovia NetFlow kolektorov, tiež ponúkajú pasívne sondy toku napojené na hlavne uzly spojenia. Tieto sondy potom vytvárajú záznamy o toku, ktorý zachytia.[10] Medzi takéto dedikované senzory a sondy môžeme zaradiť aj dnes používané riešenia ako **SPAN(Switch Port Analyzer)** alebo **TAP(Test Access point)**. **SPAN** je technológia, ktorá zrkadlí a kopíruje tok prechádzajúci prepínačom z jedného portu na iný port, kde je pripojená NetFlow sonda, z kotorej sa tieto pakety odosielajú na analýzu. V prípade vyťaženia prepínača, tok odosielaný na analýzu nemá prioritu a preto môže dojsť k jeho zahodeniu, preto aj priamo Cisco odporúča používať SPAN len v menej vyťažených sieťach. **TAP(Test Access Point)** je to hardvérové riešenie, ktoré posiela oba smery tokov na dvoch dedikovaných kanáloch spojených so sondami, vyžiadané dáta prichádzajú na analýzu de facto v reálnom čase. Dedikované sondy sú aj častokrát lacnejšie ako samotné prepínače alebo smerovače a sú určené len pre zachytávanie toku. [11] Ukážka NetFlow architektúry s TAP aj SPAN v sietí je na obrázku 1.4
- **Virtuálne senzory toku** - Virtuálne sondy dedikované vo virtuálnom prostredí, ktoré monitorujú dátovú prevádzku prechádzajúcu cez virtuálne prepínače, následné túto prevádzku zachytávajú, vytvárajú z nej tok a exportujú ju na kolektor.[10]

NetFlow proces na exportérovi funguje na princípe, že pre každý IP tok, ktorý ním prechádza zaznamená čas vzniku, čas ukončenia, počet paketov a bajtov, prípade ďalšie informácie v závislosti od verzie protokolu NetFlow. Exportér exportuje dáta o tokoch na kolektor po uplynutí nejakého času prípadne ak potrebuje uvoľniť pamäť pre ďalšie toky. Následne sú takéto toky na exportérovi zahodené, preto ak sa počas posielania exportovaného toku na kolektor vyskytne chyba v prenose, tak exportované dáta nemusia na kolektor nikdy doraziť a tok je teda zahodený. Princíp zachytenia a exportu dát z exportéra je možné rozdeliť do nasledujúcich bodov:

1. Príjem paketových dát a extrahovanie informácií
2. Vytvorenie prípadne aktualizovanie záznamov v cache pamäti NetFlow
3. Časová exspiácia toku
4. Export NetFlow toku

Exspirácia toku je časový rámec, ktorý definuje za aký čas sa tok bude exportovať. Tok sa považuje za neaktívny ak nie sú zachytené ďalšie pakety patriace danému toku v mieste pozorovania pre daný časový limit. Ak paket dorazí v časovom limite, tak je tok stále aktívny.

Exportér posiela tok na kolektor na základe nasledujúcich podmienok:

- Ak exportér deteguje koniec toku. Príkladom môže byť FIN alebo RST bit v TCP(Transmission Control Protokol – Protokol Riadenia prenosu) spojení, tak sú záznamy o toku exportované
- Ak je tok neaktívny po určitý časový úsek. Tento úsek by mal byť nastaviteľný na *Exportérovi* s minimálnou hodnotou 0 pre okamžitú časovú exspiráciu a export.
- Pri dlhotrvajúcich tokoch by mal exportér toky exportovať na pravidelnej báze, na *Exportérovi* by mal byť časový úsek exportu konfigurovateľný.
- V prípade obmedzení exportéra napríklad nedostatok pamäte, by mal čas pre tok vypršať predčasne a tok sa exportovať.

Exportér môže tok aj vzorkovať(sampling), kedy posiela len každý N-tý paket, vzorkovanie je nastaviteľné a sú na ňom založené niektoré obdoby NetFlow protokolu, napr. *sFlow*. Vzorkovanie len niektorých paketov je menej náročné na výkon hardvéru. [7]

Kolektor

Kolektory sú zariadenia, ktoré ukladajú informácie o dátových tokoch do databáz a na pamäťové média. Záznamy o tokoch dostávajú z jedného alebo viacerých exportérov pomocou UDP spojenia, ktoré nie je spoľahlivé, z toho dôvodu môže pri výpadku spojenia dôjsť k strate toku. Kolektory majú viacero dôležitých úloh medzi ktoré patrí:

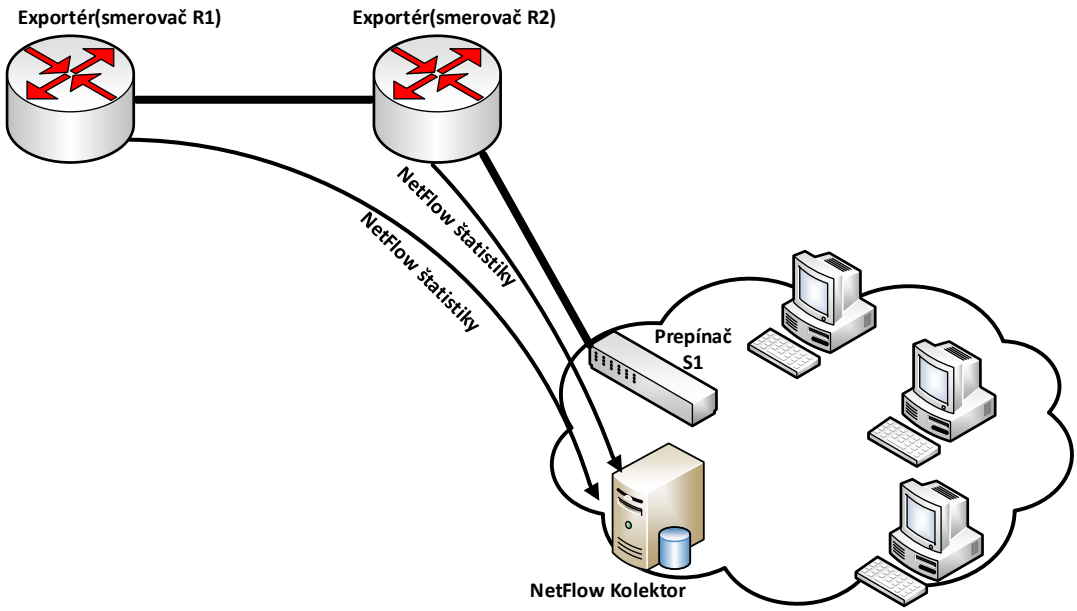
- Identifikovanie duplikácie tokov - V prípade siete, kde je viacero exportérov, môže dochádzať k duplikácií tokov a teda ukladanie rovnakého toku viackrát. Kolektor musí sledovať duplicitu tokov, prípadne duplicitné toky odmazať.
- Spájanie zrkadlených tokov - NetFlow generuje záznamy, ktoré nemajú definovaný smer, preto pre spojenia môžu existovať dva toky. Kolektor musí tieto toky vyhodnotiť a priradiť k sebe, aby reflektovali dané spojenie, príkladom na začiatku bolo spojenie HTTPS, ktoré v sebe nesie dva toky.
- Analýza správania tokov a vzorcov - Kolektory orientované na bezpečnosť musia byť pomocou algoritmov, schopné analyzovať toky pre prípadné bezpečnostné hrozby.

- Ukladanie toku - Kolektor môže tok ukladať dni, mesiace, prípade roky v závislosti na dôležitosti dát. Databáza kolektoru je následne používaná na vyšetrovanie incidentov.

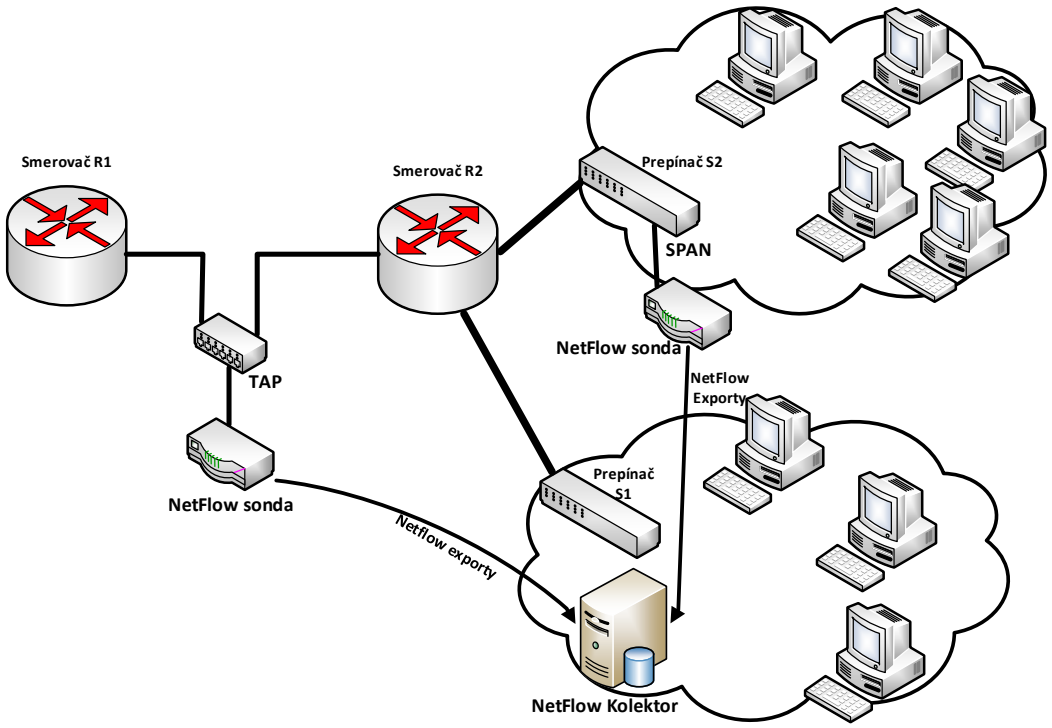
Konzola manžmentu

V prípade veľkých sietí, množstva exportérov a kolektorov je potrebné mať centrálnu manažmentovú platformu ktorá umožňuje množstvo funkcií, vyhodnocovanie a spravovanie tokov. Takáto platforma umožňuje detailnejšiu analýzu tokov, ukážku výstrah, ktoré vznikli pri zadaných podmienkach. Pomocou takejto platformy je možné spravovať bezpečnostné pravidlá, prípadne nastavovať rôzne politiky prístupov, vytvárať grafy a celkové reporty o viacerých tokoch. [10]

Tradičná architektúra NetFlow využíva ako exportéry priamo smerovače v sietí, čo je však výkonovo náročné na výpočový výkon smerovačov.



Obr. 1.3: Tradičná architektúra NetFlow - export pomocou smerovačov



Obr. 1.4: Moderná architektúra NetFlow - exporty pomocou sond, zachytávanie dátového toku pomocou technológií SPAN a TAP

1.5.3 Záznamy NetFlow v9

V samotnom protokole NetFlow sa počas vývoja menili formáty pre správy, ktoré protokoly rôznych verzií podporovali. Pridávali sa nové polia do hlavičiek a tela správ, ktoré sa rozširovali o informácie v závislosti od verzie. Zatiaľ najnovšia verzia 9 NetFlow protokolu prináša zásadný rozdiel oproti predchádzajúcim verziám a to tým, že je založená na šablónach. Šablóny umožňujú rozšírenie záznamov a ich informácií a tým vytvárajú priestor pre úpravu záznamu protokolu aj do budúcnosti, bez potreby meniť základný formát správ. Výhody takejto implementácie sú:

- Aplikačné prispôsobenie vo firemnej sfére. Firmy nebudú nútené pretvárať svoje aplikácie aby používali už zadefinované formáty správ, ale môžu definovať svoje šablóny pre aplikácie a ich už existujúce alebo nové funkcie.
- Pridanie nových funkcií a informácií do záznamov je rýchle a efektívne. Nevyžaduje meniť súčasnú implementáciu NetFlow v9.
- Formát verzie 9 používajú ako inšpiráciu ďalšie široko implementované protokoly a môže byť využitý ako základ pre ďalšie vyvíjajúce sa protokoly.

Štruktúra záznamu v9 je rozsiahla, preto je potrebná definicia pojmov, ktoré sa vyskytujú pri NetFlow verzií 9:

Exportovaný paket

Paket pochádzajúci od exportéra, ktorý zachytáva záznamy o toku. Tento paket je smerovaný na NetFlow kolektor. Ukážka exportovaného paketu so šablónami FlowSet je na obrázku 1.5

Packet header

Je to prvá časť exportného paketu. Hlavička paketu poskytuje základné informácie o pakete - NetFlow verziu, počet záznamov uložených do paketu a číslo sekvencie.

Šablóna záznamu

Definuje formát pre ďalšie dáta záznamu. Tieto dáta podľa preddefinovaného formátu môžu byť obsiahnuté v súčastnom alebo nasledujúcom exportnom pakete. Nevyhnutne nie je potrebné aby bol záznam šablóny obsiahnutý v každom exportnom pakete, avšak NetFlow kolektor musí ukladať záznam šablóny na interpretáciu súvisiaceho dátového toku, ktorý obdrží v následujúcich paketoch toku. Tieto dátové záznamy následné spojí so skôr prijatou šablónou záznamu.

Identifikátor šablóny

Unikátne číslo šablóny, používane na odlíšenie jednotlivých šablón záznamu, ktoré vzniknú na rovnakom exportérovi, teda je zabezpečená lokálna unikátnosť na sledovacom bode. Nie je teda zabezpečená unikátnosť pre aplikáciu kolektoru, ktorá v prípade, že zachytáva exportovaní pakety z viacerých exportérov, by mala byť upozornená. Kolektor by si v tomto prípade mal v poradí ukladať do pamäte adresy jednotlivých zariadení, ktoré vytvárajú exportné pakety a identifikátory šablón, pre zaistenie unikátnosti.

FlowSet

Termín označujúci kolekciu jedného alebo viacerých záznamov o toku, ktoré majú podobnú štruktúru. V exportnom pakete jeden alebo viacero FlowSet polí nasleduje po hlavičke paketu obr.1.5. Existujú tri typy FlowSet polí - *FlowSet šablóny*, *Voliteľná šablóna FlowSet* a *Dátový FlowSet*. [7]

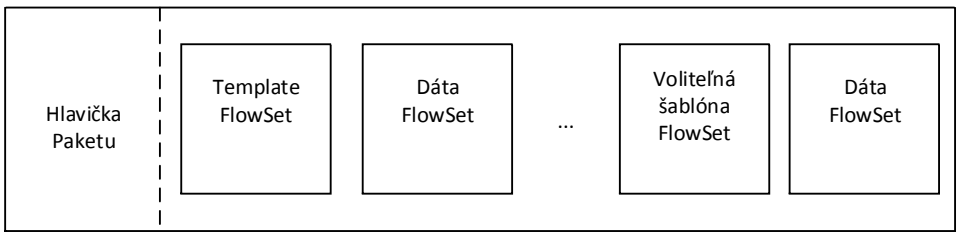
Záznam dát

Poskytuje informácie o IP toku zachytenom na sledovacom bode. Skupiny záznamov dát referujú predchádzajúce identifikačné čísla šablón. Tieto referencie slúžia na rozdelenie dát v zázname.

1.5.4 Štruktúra paketu NetFlow v9

Každý záznam na začiatku obsahuje hlavičku(Packet Header), po hlavičke nasleduje minimálne jedna alebo viacero šablón(template), potom nasledujú dáta uložené podľa danej šablóny. Exportovaný paket môže mať tieto podoby:

- Exportovaný paket bude pozostávať zo šablóny a Dát FlowSet. U kolektora by nemalo nastať, že definovaný identifikátor šablóny v danom pakete bude mať špecifickú príbuznosť len s dátami FlowSet obsiahnutými v danom pakete. Kolektor si musí ukladať každú šablónu a priradiť jej správne identifikačné čísla na správnu interpretáciu dátových záznamov.
- Exportovaný paket bude obsahovať najmä FlowSet dáta. Ak dôjde k správnemu zaslaniu a definovaniu identifikačných čísiel šablón na kolektor, bude väčšina paketov pozostávať z predpísaných FlowSet dát.
- Exportovaný paket bude obsahovať len šablóny FlowSet. Využíva sa to v prípade, že dôjde k reštartu exportéru a následne je potrebná synchronizácia s kolektorom. Exportér teda odosiela šablóny FlowSet na to aby zabezpečil, že kolektor bude mať informácie na interpretáciu každého záznamu FlowSet dát. Šablóny záznamov musia byť periodicky z dôvodu obmedzeného času platnosti, ak v tomto čase obnovovania nie je na exportérovi žiadny záznam FlowSet dát, ktorý je potrebné odoslať na kolektor, tak sa odošle exportovaný paket so šablónami. [12]



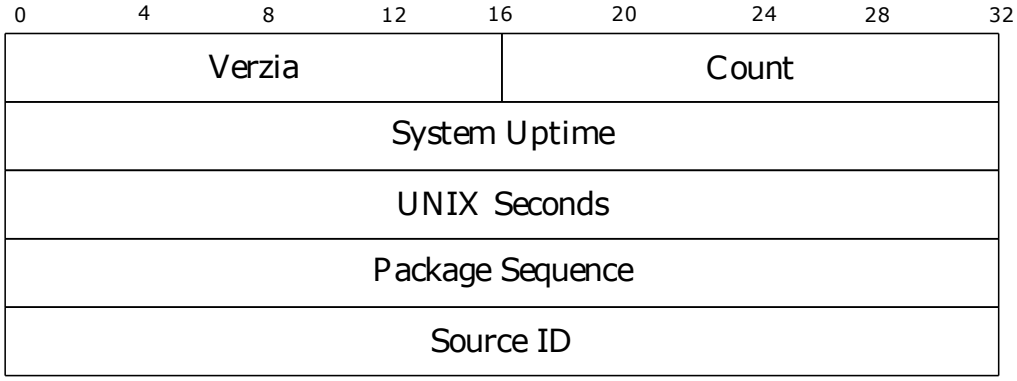
Obr. 1.5: Ukážka príkladu exportovaného paketu NetFlow v9 s FlowSet poliami

Obsah	Popis
Packet Header	Hlavička paketu
Template FlowSet	Šablóna FlowSet
Data FlowSet	Dáta uložené podľa šablóny
Data FlowSet	Dáta uložené podľa šablóny
...	...
Template FlowSet	Šablóna FlowSet
Data FlowSet	Dáta uložené podľa šablóny
...	...

Tab. 1.2: Formát štruktúry exportovaného paketu a popis polí

Hlavička paketu NetFlow v9

Hlavička paketu NetFlow v9 sa veľmi nelíši od predchádzajúcich verzií. Formát ostal relatívne nezmenený a čerpá predlohu z formátu verzie 5. Začiatok hlavičky tvoria informácie o verzií, následne počet šablón a dát, potom nasledujú časové značky a ďalšie informácie.[12] Ukážka hlavičky je na obrázku 1.6, popis ďalších polí hlavičky v tab.1.3



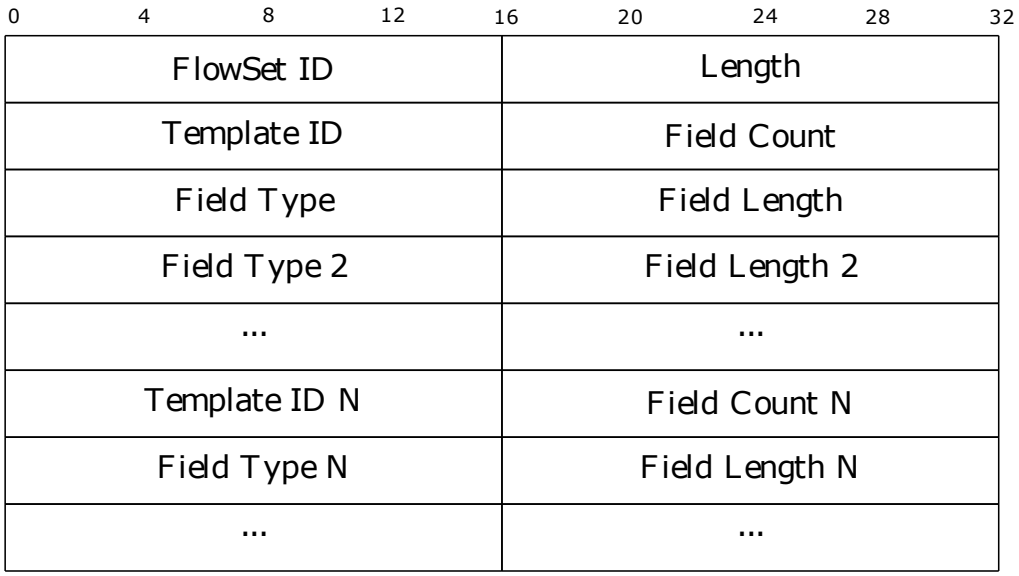
Obr. 1.6: Ukážka hlavičky exportovaného paketu NetFlow v9

Bajty	Obsah	Popis
0-1	Version	Verzia NetFlow záznamu, verzia 9 má hodnotu 0x0009
2-3	Count	Počet FlowSet záznamov v pakete
4-7	System Uptime	Čas v milisekundách odkedy je zapnutý exportér
8-11	UNIX Seconds	Aktuálny čas v sekundách od roku 1970
12-15	Package sequence	Sekvenčné číslo všetkých exportovaných paketov, čo poslal exportér. Slúži na identifikácie chýbajúceho paketu
16-19	Source ID	Hodnota, ktorá určuje garanciu unikátnosti pre každý tok exportovaný z konkrétneho zariadenia

Tab. 1.3: Popis hlavičky paketu NetFlow 9

Šablóna FlowSet a jej štruktúra

Jedným zo základných prvkov v NetFlow v9 protokole je šablóna FlowSet. Umožňuje a zlepšuje flexibilitu formátu NetFlow záznamu, pretože kolektoru alebo aplikačnému rozhraniu dáva možnosť spracovať dátové záznamy aj bez nevyhnutnej vedomosti poznať interpretáciu všetkých dát v dátovom zázname.



Obr. 1.7: Formát a veľkosť(v bajtoch) jednotlivých polí v šablóne FlowSet

FlowSet ID	Identifikačné číslo záznamu
Length	Veľkosť FlowSet
Template ID	Identifikačné číslo šablóny
Field count	Počet polí
Field type	Typ prvého poľa
Field length	Dĺžka prvého poľa
Field type	Typ druhého poľa
Field length	Dĺžka druhého poľa
...	...
Template ID N	Identifikačné číslo N-tej šablóny
Field count	Počet polí N-tej šablóny

Tab. 1.4: Popis štruktúry FlowSet šablóny

Identifikačné číslo záznamu sa používa na odlíšenie šablóny záznamov od záznamov dát. Identifikačné číslo šablóny záznamov je v rozsahu 0-255. Je zaznamenaná celková veľkosť FlowSet a to, z dôvodu že aj individuálna FlowSet šablóna môže obsahovať viacero čísiel ktoré identifikujú šablóny. Pozíciu nasledujúceho FlowSet záznamu určuje práve veľkosti, nasledujúci FlowSet záznam môže byť buď šablóna alebo priamo dáta FlowSet. Pomocou identifikačného čísla šablóny sa identifikuje, ktorá šablóna sa použije na záznamy dát.

Šablóna FlowSet môže obsahovať viacero záznamov šablón, preto počet polí viazaných k danej šablóne určuje začiatok a koniec nového záznamu šablóny. Dĺžka polí(Field Length) určuje veľkosť jednotlivých typov polí v bajtoch. NetFlow v9 podporuje 79 typov polí(ukážka v prílohe), ktoré môžu byť obsiahnuté v šablóne, avšak, nie všetky typy musí nevyhnutne podporovať exportér, v takom prípade ich nebude zaznamenávať. Tých 79 možných polí môže v sebe niesť veľké množstvo ďalších informácií o dianí v sietí. V prípade potreby rozšíriť šablónu, je potrebné zadefinovať nový typ poľa do zoznamu. Takýto typ poľa musí byť aktualizovaný v zázname ako v exportérovi, tak aj na strane kolektoru, avšak exportovaný formát NetFlow by sa nemal meniť. U niektorých polí je veľkosť poľa nememnná a preddefinovaná napevno, napríklad protokol alebo cieľová, alebo zdrojová adresa IPv4(obr. 1.8), rovnako sú obsiahnuté polia aj s premennou veľkosťou, ktoré umožňujú efektívnejšie využívanie miesta na kolektory a majú menší dopad na priepustnosť siete. [7]

V prípade potreby je možné definovať si aj vlastnú šablónu FlowSet formátu, ukážka obr.1.5. Ktorá sa zväčša používa k prenosu informácií o špecifických NetFlow dátach, prípadne procese a IP toku. Je možné použiť vzorkovanie(sampling) dát.

Príkladom môže byť použitie takejto voliteľnej šablóny FlowSet na zaznamenávanie vzorkovania pre špecifické rozhrania. [7]

FlowSet ID	Identifikačné číslo záznamu, stále 1, odlišuje šablónu od dát.
Length	Veľkosť daného záznamu
Template ID	Identifikačné číslo šablóny, vygenerované exportérom
Option scope length	Dĺžka každého poľa v tejto šablóne
Option Length	Dĺžka obsahu poľa v tejto šablóne
Scope N field type	Určenie relevantnej časti do ktorej patrí daný záznam
Scope N field length	Dĺžka relevantnej oblasti poľa
Option filed N type	Typ poľa
Option filed N length	Dĺžka poľa
Padding	Zarovnanie/Vyplnenie záznamu na 4 bajty(32bitov)

Tab. 1.5: Štruktúra vlastnej šablóny FlowSet

Štruktúra formátu dát FlowSet

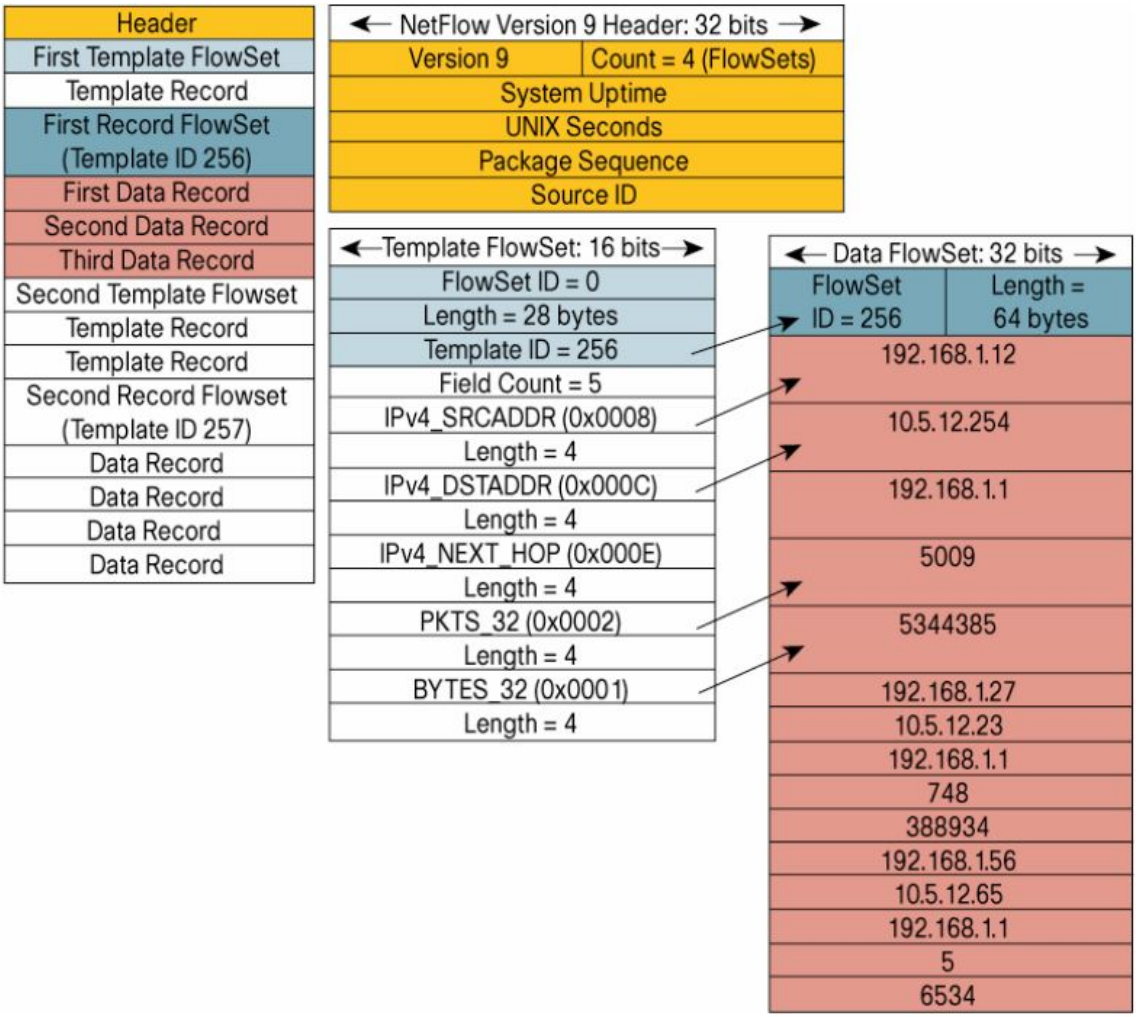
Každý dátový FlowSet musí byť spojený s identifikátorom FlowSet šablóny. Štruktúra týchto dát je teda závislá na šablóne, ktorá určuje ako budú dáta ukladané.

V prípade, že sa nenájde šablóna pre FlowSet dáta, tak by sa mal záznam týchto dát zahodiť. Každý zo záznamov obsahuje niekoľko polí. Typ polí a ich veľkosť definovala predtým šablóna. [7]

FlowSet ID=Template ID	Identifikačné číslo, používa sa na párovanie dát s FlowSet šablónou
Length	Veľkosť celého záznamu FlowSet dát
Record 1 - Field 1	Záznam 1 - Pole 1
Record 1 - Field 2	Záznam 1 - Pole 2
Record 1 - Field 3	Záznam 1 - Pole 3
...	...
Record 2 - Field N	Záznam 2 - Pole N
Record 3 - Field N	Záznam 3 - Pole N
...	...
Padding	Zarovnanie/Vyplnenie záznamu na 4 bajty(32bitov)

Tab. 1.6: Štruktúra šablóny FlowSet Data a jej záznamy

Na nasledujúcom poslednom obrázku môžeme vidieť konkrétne exportovaný paket s naplnenými poliami a identifikátormi šablón.



Obr. 1.8: Príklad exportovaného paketu a ukážka jednotlivých polí s konkrétnymi poliami a dátami v nich, presne podľa definície flow record formátu firmy Cisco [13]

1.6 IPFIX protokol

IPFIX protokol (IP Flow Information Export – Informačné exporty o IP toku) je jednosmerný IETF protokol určený na export dát a informácií o IP toku. Na prenos informácií o IP toku používa dáta zo smerovačov, sieťových sond a ďalších zariadení, ktoré sprostredkuvávajú služby sieťového manažmentu, účtovné a fakturačné služby alebo vykonávajú meranie prevádzky na sieti. Tento protokol prináša kompaktnosť a ucelenosť riešení k zhromažďovaní informácií o sieťovom toku a určuje postupy tohto procesu. Bol vyvíjaný od začiatku ako vysoko škálovateľný informačný protokol, obsahujúci polia a šablóny u ktorý je možná vlastná definícia prenášaných dát.

Protokol IPFIX čo sa týka architektúry zariadení je založený na rovnakom princípe ako protokol NetFlow v9 z ktorého primárne vychádza a teda obsahuje exportéri, kolektory, sondy, sledovacie body a na konci analyzátory. Avšak architektúra procesov je rozsiahlejšia a obsahuje ďalšie položky.[14]

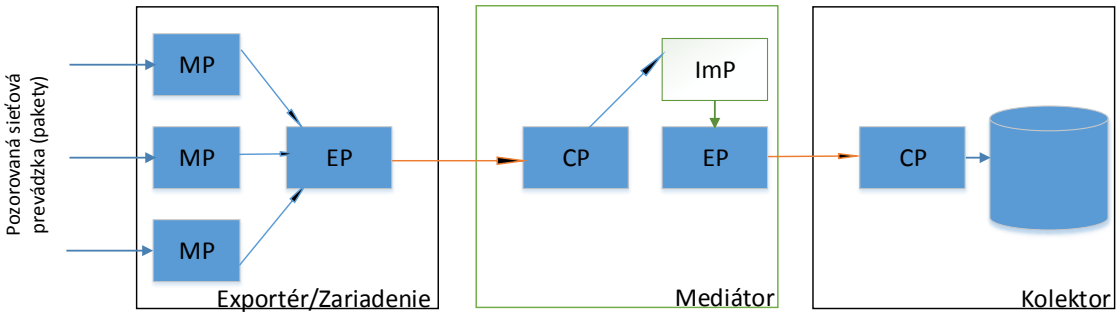
1.6.1 Architektúra Procesov IPFIX

- Proces Merania(Metering Process - MP) - tento proces generuje záznamy o dátovom toku z paketov, ktoré zachytí na sledovacom bode. Tento proces vykoná zachytenie paketu, pridá mu časovú značku, paket navzorkuje a klasifikuje takto dátový tok. Rovnako udržiava tok v interných dátových štruktúrach. Takýto tok následne posiela na proces exportu.
- Proces Exportu(Exporting Process - EP) - pomocou tohto procesu sú dátové toky z jedného alebo viacerých procesov merania posielané pomocou IPFIX na kolektor na proces zhromažďovania.
- Proces zhromažďovania(Collection Process - CP) - je to proces prijímania záznamov o toku cez IPFIX z jedného alebo viacerých procesov exportu, exportérov.

V IPFIX protokole existuje ešte ďalšie zariadenie, ktoré sa volá *Mediátor*. Mediátor zbiera, transformuje a preposiela IPFIX toky správ. Na mediátore sa využívajú tzv. intermediačné procesy(Intermediate Processes - ImP), ktoré v sebe zahŕňajú radu úkonov [15]:

- *Anonymizáciu*
- *Agregáciu*
- *Filtrovanie, zastúpenie(proxying), multiplexovanie/demultiplexovanie dát*
- *Preklad protokolov a iné*

Ukážka zariadení v architektúre a procesov na obrázku 1.9



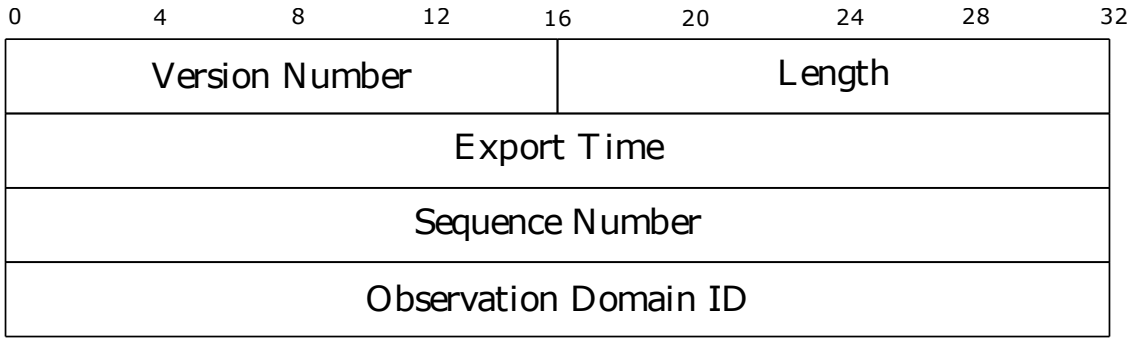
Obr. 1.9: Ukážka architektúry procesov na zariadeniach protokolu IPFIX

1.6.2 Štruktúry protokolu IPFIX

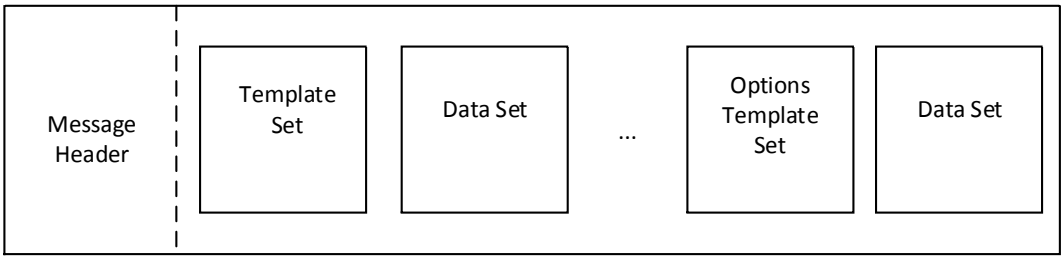
IPFIX posiela dáta pomocou správ. Správy obsahujú *hlavičku správy(Message Header)* a jeden alebo viacero *Setov*. Set obsahuje podobne ako celá správa hlavičku(Set Header), ktorý môže pozostávať z troch možností:

1. Setu šablóny(Template Set), ktorá obsahuje záznamy šablóny(Template Records)
2. Voliteľného setu šablóny(Options Template Set), ktorá obsahuje voliteľné záznamy šablóny
3. Dátového setu(Data Set), ktorý obsahuje priamo dátové záznamy

Tu môžeme vidieť, že štruktúra správy je dosť podobná správe NetFlow v9. Mení sa však rozsah jednotlivých štruktúr a ich názvy.[15]



Obr. 1.10: Hlavička správy IPFIX a veľkosti jednotlivých polí



Obr. 1.11: Príklad správy IPFIX

Správa IPFIX môže mať ďalšie podoby:

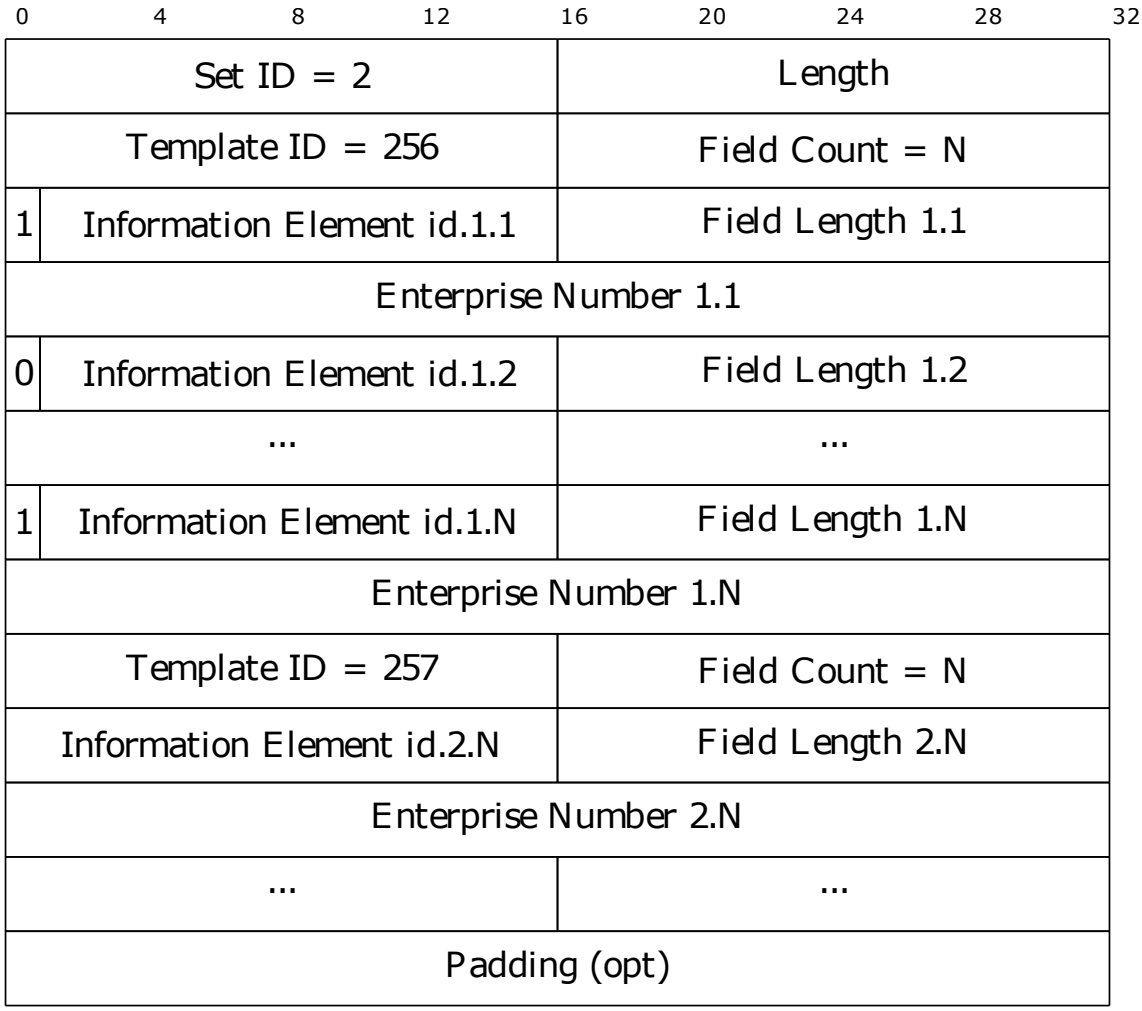
- Správa bude pozostávať len z hlavičky a dátových setov(Data sets), použije sa vtedy keď už boli definované príslušné záznamy šablón na proces zhromažďovania(Collecting Process)
- Správa bude pozostávať len zo setov šablón(Template set) a voliteľných setov šablón(Options Template set). Takáto správa sa používa na definovanie alebo opakované definovanie šablón a voliteľných šablón, napríklad po periodickom vypršaní šablón, alebo pri reštarte kolektora, ktorý nemá v tú chvíľu dané šablóny.

Šablóny a Informačné Prvky(Templates & Information Elements)

Šablóny definujú štruktúru záznamov dát v rámci súboru dát(Data Set), sú definované identifikátorom šablóny(Template ID), ktorá korešponduje s identifikátorom setu, hlavičkou tohto setu a dátami setu. Šablóny pozostávajú z párov(informačný prvok - IE a dĺžky).

Informačné prvky(Information Elements) poskytujú informácie o type poľa samotnej šablóny. Informačné prvky sú vlastne najpoužívanejšie prípady využívania tokov, definované v IANA. Napríklad tradičný päťprvkový tok(IP adresy, porty, protokol), ošetrenie paketov(adresa nasledujúceho smerovača, cieľové číslo bgp AS), časové značky(začiatok toku v sekundách, zachytený čas v mikrosekundách a iné), hlavičky IPv4, IPv6, ICMP, UDP(čísla sekvencií, ipTTL), je ich definovaných IANA okolo 400.

Na rozšírenie vlastnosti informačných prvkov sa používa ešte priradené číslo Enterprise Number. Pomocou tohto čísla môžeme exportovať informácie o type informačného prvku definovaného IANA.[15]



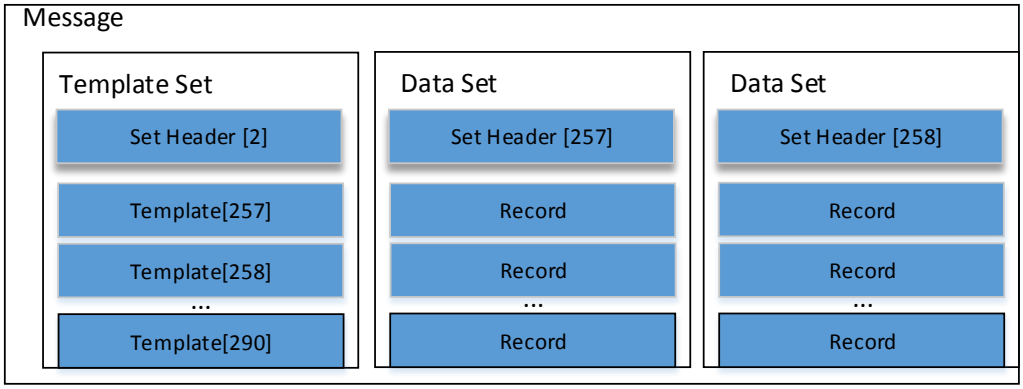
Obr. 1.12: Formát a veľkosť(v bajtoch) jednotlivých polí v šablóne Set IPFIX

Formát Dátových Záznamov (Data Set Records)

Záznamy dát sú posielané v dátových setoch. Dátové záznamy pozostávajú z jedného alebo viacerých polí hodnôt(Field Value). Kód šablóny, do ktorej patria dané hodnoty polí, je zakódovaný v Set Header(Hlavička setu) v polí "Set ID", kde Set ID = Template ID, teda identifikátor šablóny sa zhoduje s daným setom, pod ktorý patria záznamy s jeho identifikátorom.[15] Ukážku správy Data Set s jednotlivými poliami ich veľkosťami a záznamami na obr. 1.13
Formát celej správy obsahujúcej ako šablóny, tak aj dátové záznamy data set na obr.1.14

0 4 8 12	16 20 24 28 32
Set ID = Template ID	Length
Record - 1 - Field Value 1	Record - 1 - Field Value 2
Record - 1 - Field Value 3	...
Record - 2 - Field Value 1	Record - 2 - Field Value 2
Record - 2 - Field Value 3	...
Record - 3 - Field Value 1	Record - 3 - Field Value 2
Record - 3 - Field Value 3	...
...	Padding (opt)

Obr. 1.13: Formát a veľkosť jednotlivých polí v šablóne záznamov Data Set IPFIX



Obr. 1.14: Ukážka celkovej štruktúry správy IPFIX - šablóny a dátové sety k ním

Hlavné rozdiely a vylepšenia IPFIX oproti NetFlow v9 protokolu

- IPFIX je štandardizovaný protokol, NetFlow v9 je proprietárny protokol spoločnosti Cisco(a používaný ďalšou radou výrobcov)
- IPFIX protokol obsahuje približne 386 typov polí zahrnutých v šablóne, NetFlow v9 približne 79
- Obidva protokoly majú rozšíriteľné šablóny
- IPFIX obsahuje informačné prvky ktoré sú definované pre mnoho výrobcov, NetFlow v9 obsahuje informačné prvky spoločnosti Cisco
- IPFIX obsahuje rozšírené informácie o informačných prvkoch pomocou Enterprise elements(podnikové prvky), NetFlow v9 túto možnosť nemá
- Obidva protokoly majú aj polia s premennou dĺžkou polí
- Obidva protokoly majú štrukturované dáta
- Transportný protokol u IPFIX môže byť: TCP, UDP a SCTP, pri NetFlow v9 je to: prioritne UDP, SCTP(len pre Cisco)
- U IPFIX je zabezpečená bezpečnosť dát o toku jeho integritou a šifrovaním, využíva sa: TLS(Transport Layer Security) pre TCP prenos alebo DTLS(Datagram Transport Layer Security) pri prenosoch UDP a SCTP. U NetFlow v9 neexistuje zabezpečenie záznamov o toku, dáta je možné odchytiť v exportovanej podobe.
- Časovače pre cache pamäť sú pri IPFIX nastaviteľné, u NetFlow v9 sú fixne dané
- IPFIX štandardizovane podporuje - IPv6, MPLS(Multiprotocol Layer Switching), VLAN(Virtual Lan), MAC adresy, Multi-cast a IPsec tuneling. NetFlow v9 podporuje tieto funkcie a technológie len pre Cisco zariadenia a zahrnutých výrobcov
- Oba protokoly sú v podstate jednosmerné, čo sa týka čistého exportu o dátovom toku, avšak môžu v istých ohľadoch niesť aj tok opačným smerom, kedy nesú informácie o zariadeniach na exportér. Pri NetFlow v9 to funguje len pre zariadenia Cisco, avšak u IPFIX môže dochádzať k prenosu informácií o informačných prvkoch pomocou súkromného podnikového čísla(Enterprise Private number) [16]

1.7 sFlow, NetStream, OpenFlow protokoly

Táto kapitola sa skrátene venuje ďalším informačným protokolom o dátovom toku.

sFlow protokol

sFlow(sampled Flow) je vzorkovací protokol toku, ktorý je štandardom na export paketov na druhej vrstve OSI modelu. O údržbu protokolu sa stará konzorcium sFlow.org, ktoré vydáva aj jeho špecifikácie. Súčasná verzia je v5.

sFlow protokol používa povinné vzorkovanie na zachovanie elasticity systému, práve pre tento dôvod je používaný vo veľmi rýchlych sieťach(rádovo niekoľko Gigabitov za sekundu). V podstate sa nejedná o protokol, ktorý exportuje celý tok ale exportuje len určitý počet paketov v čase, kľudne aj z viacerých tokov, čo síce môže mať dopad na presnosť avšak z hľadiska dlhšej doby je to dostatočná presnosť.

Používa podobne ako NetFlow protokol exportéry na ktorých bežia tzv. počítadlá vzoriek(Counter samples), ktoré následne posielajú dané počítadlá na kolektor. Je to výhodne aj na monitorovanie množstva rozhraní.

Protokol používa na prenos vzoriek UDP prenosový protokol a vytvára sFlow datagramy.[17]

Využitia funkcií sFlow dát je podobné ako pri NetFlow protokole, napríklad:

- Detegovanie, diagnostika a riešenie problémov na sieti
- Správa preťaženia siete v reálnom čase
- Pochopenie aplikačného mixu a ich zmien (Web, DNS ...)
- Účtovanie a fakturácia služieb
- Zlepšovanie smerovania a optimalizovanie prepojenia
- Plánovanie kapacít

Technológia vzorkovania sFlow spĺňa kľúčové požiadavky na monitorovanie dátového toku - je to **priemyselný štandard** podporovaný implementáciou medzi dodávateľmi, **poskytuje široký pohľad na sieť** a jej zaťaženie, **je škálovateľný** s podporou až do rýchlosti 10Gb/s bez dopadu na výkon hlavných smerovačov a prepínačov, a v neposlednom rade je to **nízko-nákladové riešnie** na použite od malých skupín až po chrbticové hlavné smerovače, bez dopadu na celkový výkon alebo pamäť.[17]

NetStream Protokol

Proprietárny prtokol spoločnosti Huawei. Je založený na sledoch(streamoch) dát. Využíva podobné záznamy ako NetFlow protokol, rovnako architektúra sa skladá z NDE (NetStream Data Exporter) - exportéru, NSC(NetStream Collector) - kolektoru a NDA (NetStream Data Analyzer) - analyzeru a role týchto zariadení sú rovnaké ako v prípade NetFlow. Avšak, NetStream protokol môže byť použitý aj

na meranie ATM, POS portov, prípadne rozhrania vo VPN. Vo verzií 9 používa rovnako ako NetFlow v9 šablóny a formát správ je veľmi podobný, využíva rovnako UDP datagramy. Dnes sú podporované tri verzie: V5, V8 a V9. Oproti NetFlow však Huawei definoval viacero typov ich vlastných proprietárnych polí v správach a voliteľných dát, ktoré môže protokol niesť.

Štruktúra šablóny stream data a následne dát v streame je len ľahko zjednodušená oproti NetFlow šablónam. Rozdiel oproti NetFlow predstavuje už implementovaný NetStream Data Analyzér, kde sa posielajú dáta z NetStream Collectoru, a na danom analyzéry beží grafické rozhranie, pomocou ktorého je jednoduché analyzovať dáta zachytené z NSC. Výstup dát z exportéru je pri NetStream protokole možný troma spôsobmi: tok za tokom(Stream-by-stream), vzorkovanie(sampling) a agregácia.[18] Podobne ako pri iných informačných protokoloch, je pre NetStream typické aplikačné využitie napr. :

- Nasadenie NDE(NetStream Data exporter) - na prístupových, konvergenčných alebo chrbticových vrstvách na využitie analýzy, aplikačný monitoring a podobne.
- Bezpečnostný monitoring
- Plánovanie domén autonómnych systémov AS
- Meranie a plánovanie multicastových dátových tokov - VoD, IPTv, stream media
- Vyrovnanie medzi ISP poskytovateľmi na základe sieťovej prevádzky[18]

OpenFlow Protkol

Je to štandardizovaný protokol pre SDN(Software Defined Networks – Sofvérovo definované siete). OpenFlow umožňuje poňať smerovače a prepínače ako systém na transport dát, využíva k tomu záznamy o tokoch na týchto zariadeniach a určuje ako s danými tokmi a dátami nakladať pomocou centrálneho API na nejakom kontroléry. Takto oddelí dátovú logiku od kontrolnej a centralizuje ju pre mnoho zariadení. Umožňuje tak vytvárať presnejšie cesty cez sieťové prepínače a uľahčuje rozhodovanie pre smerovanie.

Je to protokol druhej vrstvy OSI modelu, ktorý poskytuje prístup k rovine preposielania sieťového prepínača alebo smerovača. OpenFLow v podstate definuje komunikáciu medzi OpenFlow kontrolérom a OpenFlow prepínačom, kde táto komunikácia pozostáva zo súboru správ medzi kontrolérom a prepínačom a naopak. Tieto OpenFlow správy, tak umožňujú kontroléru programovať prepínač tak aby bola možná detailna kontrola nad prepínaním a smerovaním užívateľskej prevádzky. Táto prevádzka môže vytvárať dátový tok - flow, medzi jednotlivými koncovými bodmi siete, ako TCP/UDP páry portov, VLAN bodmi, koncovými tunelovými

bodmi 3.vrstvy prípadne vstupnými portami. OpenFlow bol v začiatkoch vývoja považovaný za experimentálnu platformu, avšak komunita vývojarov sa stále stará o zabezpečenie interoperability medzi rôznymi verziami. [19]

Dnes sa OpenFlow čoraz viac nasadzuje v komerčnej sfére SDN sietí pre svoje benefity ako napríklad : umožňuje inovovanie programovaním a zavádzanie nových funkcií, prípadne odlíšenie a definovanie služieb.
Zároveň centrálna inteligencia poskytuje:

- Jednoduché poskytovanie služieb
- Optimalizáciu výkonu
- Detaline rozdelený manažment politík

Po originálnom vývoji, ktorý začal na Stanfordskej Univerzite v roku 2008, ho dnes podporujú viaceré spoločnosti vyrábajúce sieťový hardware a implementujú ho do svojich zariadení, ako napríklad firmy Cisco a Brocade, ktoré ponúkajú OpenFlow dostupné kontroléry ako Cisco XNC alebo Brocade Vyatta Controller. [20]

2 Praktická časť práce

2.1 Konfigurácia smerovača a prvotný záchytný test

2.1.1 Konfigurácia NetFlow v9 na fyzickom smerovači Cisco

V praktickej časti sa v prvom rade venujem konfigurácií smerovača Cisco umiestneného v Laborátóriu Transportních síti SC 5.35 na fakulte. Tento smerovač sme využili ako reálny exportér NetFlow v9 paketov, obsahujúce dáta a šablóny, ktoré som neskôr podrobili analýze a parsovaniu vo vytvorenom programe, ktorý bol napísaný v jazyku Python. Na konfiguráciu smerovača som využíval oficiálne datasheety a konfiguračné návody do spoločnosti Cisco. [21].

Bol použitý smerovač Cisco 1812, ktorý mal na porte FastEthernet 0 nakonfigurovanú adresu 10.0.0.81, cez ktorú sme sa pomocou SSH(Secure Shell - port 22) naň aj pripájali.

Základná konfigurácia Cisco NetFlow smerovača

```
NetFlowRouter> enable
NetFlowRouter# configure terminal
NetFlowRouter(config)# interface FastEthernet0
NetFlowRouter(config-if)# ip address 10.0.0.81 255.255.255.0
NetFlowRouter(config-if)# ip flow ingress
NetFlowRouter(config-if)# ip flow egress
NetFlowRouter(config-if)# duplex auto
NetFlowRouter(config-if)# speed auto
```

Vo výpise základnej konfigurácie smerovača môžeme vidieť priradenú IP adresu na porte a sieťovú masku. Následné používam príkazy *ip flow ingress* a *ip flow egress*, na zachytávanie prevádzky na port a zároveň aj z portu. Na porte som nastavil duplexnú prevádzku(popis v sekcií 1.2), nastavil automatickú rýchlosť portu, to znamená, že port si prispôsobí a vyjedná rýchlosť podľa pripojeného zariadenia, jeho možnosti alebo prenosového média.

Konfigurácia protokolu NetFlow v9 na smerovači Cisco 1812

```
NetFlowRouter# configure terminal
NetFlowRouter(config)# ip flow-export destination 10.0.0.20 4711
NetFlowRouter(config)# ip flow-export version 9
NetFlowRouter(config)# ip flow-export template refresh-rate 15
NetFlowRouter(config)# ip flow-export template timeout-rate 15
NetFlowRouter(config)# end
```

V konfigurácií NetFlow protokole na smerovači sme najprv definovali IP adresu (10.0.0.81) a port(4711), príkazom *ip flow-export destination* kde sa exportované pakety zo smerovača(v našom prípade exportéra), posielali. Následne sme definovali

verziu NetFlow, v našom prípade verziu 9, ktorá nesie a definuje šablóny(template), príkazom *ip flow-export version*. Potom som definoval po koľkých paketoch sa má znova poslať šablóna príkazom *ip flow-export template refresh-rate*, prípadne *ip flow-export template timeout-rate*, ktorý defuinuje po koľkých minútach má byť znova zaslaná šablóna.

Zobrazenie štatistík NetFlow dáta exportu na smerovači

```
NetFlowRouter#show ip flow export
Flow export v9 is enabled for main cache
  Export source and destination details :
  VRF ID : Default
    Destination(1)  10.0.0.20 (4711)
  Version 9 flow records
  12355 flows exported in 8488 udp datagrams
  0 flows failed due to lack of export packet
  1 export packets were sent up to process level
  0 export packets were dropped due to no fib
  0 export packets were dropped due to adjacency issues
  0 export packets were dropped due to fragmentation failures
  0 export packets were dropped due to encapsulation fixup failures
```

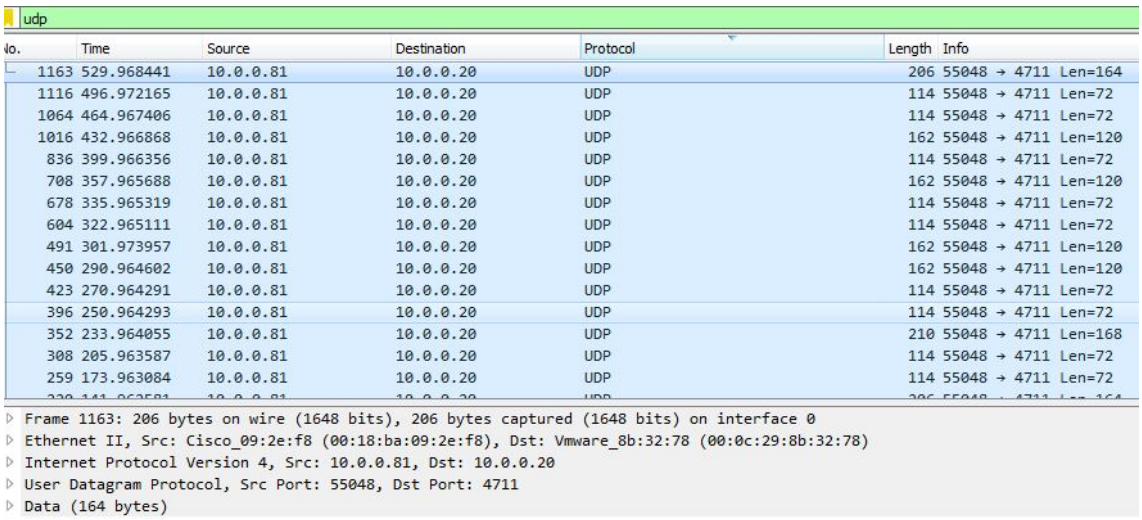
Zobrazenie detailného sumáru štatistík NetFlow

```
NetFlowRouter#show ip cache verbose flow
IP packet size distribution (62727 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
   .000 .365 .526 .004 .000 .000 .000 .025 .000 .000 .076 .000 .000 .000 .000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
  1 active, 4095 inactive, 11823 added
  551965 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 25736 bytes
  1 active, 1023 inactive, 11823 added, 11823 added to flow
  0 alloc failures, 0 force free
  1 chunk, 1 chunk added
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow     /Flow
TCP-FTP             12      0.0         2    51      0.0       0.7      15.4
TCP-WWW             12      0.0         2    50      0.0       1.0      15.1
TCP-other           96      0.0         8    83      0.0       2.5      15.0
UDP-other        11702      0.0         5    86      0.2      31.1      15.3
Total:           11822      0.0         5    86      0.2      30.8      15.3

SrcIf          SrcIPaddress    DstIf          DstIPaddress    Pr TOS Flgs  Pkts
Port Msk AS                    Port Msk AS    NextHop              B/Pk  Active
Fa0            10.0.0.20       Null           10.0.0.255      11 00  10       1
008A /0  0                     008A /24 0     0.0.0.0               229     0.0
Fa0            10.0.0.20       Local          10.0.0.81       06 00  18      18
071E /0  0                     0016 /24 0     0.0.0.0                55     2.0
```

Pomocou dvoch hore uvedených príkazov (*show ip flow export* a *show ip cache verbose flow*) sme vo výpise na stránke vyššie, zobrazili štatistiky a detailne informácie o zasielaní exportovaných paketov toku pomocou NetFlow protokolu.

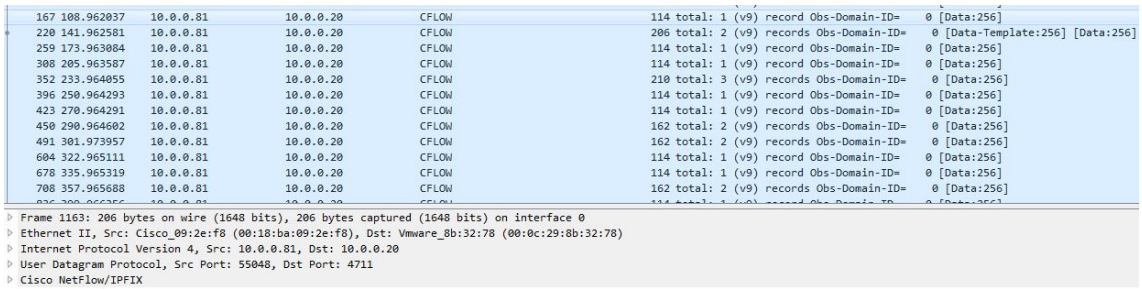
2.1.2 Overenie zasielania NetFlow paketov, naplnenie polí a šablóny v analyzačnom programe Wireshark

Na základné overenie dát a použitie šablóny sme použili sieťový analyzačný program Wireshark, ktorý je schopný zachytávať UDP prevádzku, ktorá v sebe nesie NetFlow v9 exportované pakety. Sieťovú prevádzku sme zachytávali na porte školského serveru(v našom prípade cieľová IP adresa 10.0.0.20 a port 4711), kde testovací NetFlow smerovač(zdrojová adresa IP 10.0.0.81) posielal UDP tok obsahujúci NetFlow dáta, obr. 2.1



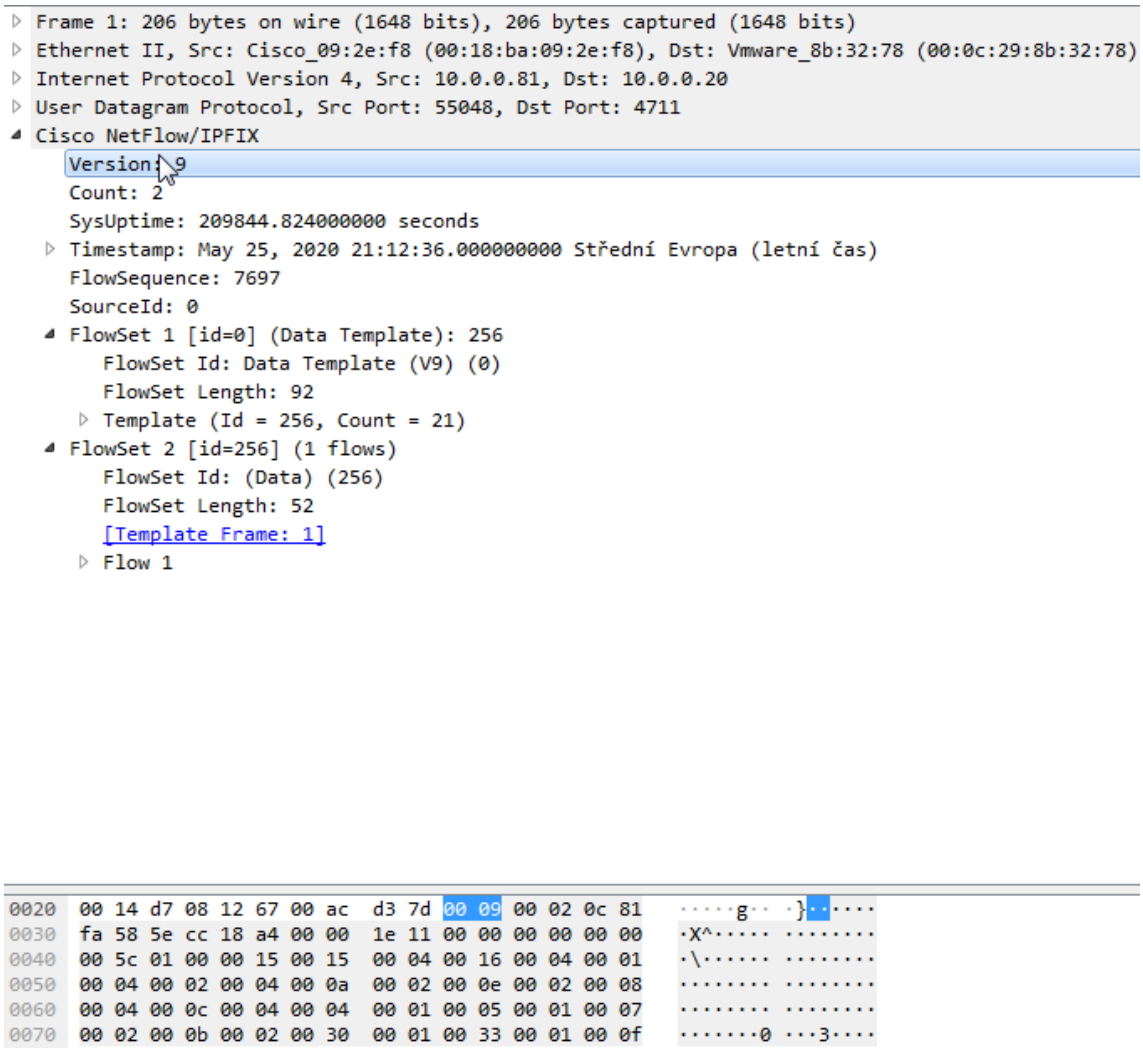
Obr. 2.1: Zachytený UDP tok z IP 10.0.0.81 na IP 10.0.0.20 port 4711

Následne sme pomocou funkcie *Decode as* vo Wiresharku, dekódovali UDP dáta ako CFLOW, ktoré odpovedajú Cisco NetFlow dátam alebo IPFIX protokolu. 2.2



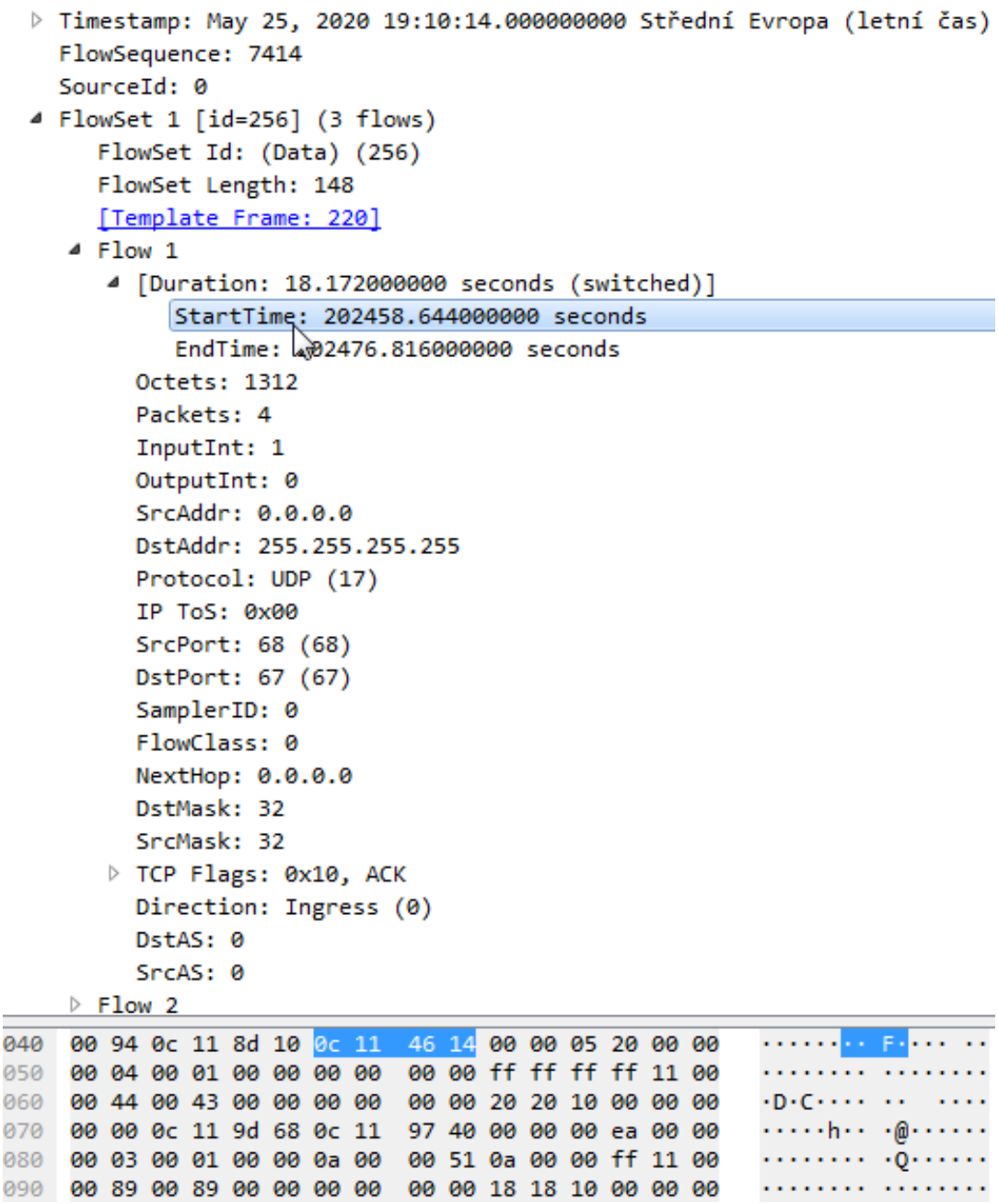
Obr. 2.2: Dekódovaný CFLOW tok z IP 10.0.0.81 na IP 10.0.0.20 port 4711

Z CFLOW toku sme skontrolovali dáta a šablónu, ktorá sa nesie v niektorých exportovaných paketoch(nie nutne v každom).



Obr. 2.3: Zachytené dáta so šablónou

Na obrázku 2.3 môžeme vidieť odoslanú šablónu, kde v identifikátor FlowSet je nastavený na 0, čo značí že v pakete sa nachádza šablóna, v našom konkrétnom prípade šablóna 256. Zároveň môžeme vidieť v hexdumpe paketu, kde sa nachádza hexadecimálne číslo ako identifikátor verzie. Použitý identifikátor bude naparsovaný do triedy Header vo vytvorenom programe, tak ako ďalšie hexadecimálne hodnoty podľa daných tried. Paket následne obsahuje FlowSet, kde je uložený tok(Flow 1), ktorý nesie dáta pre polia definované v šablóne 256. Preto sa FlowSet ID zhoduje s Template ID(identifikátorom šablóny).



Obr. 2.4: Hodnoty naplnené v jednotlivých poliach

Na obrázku 2.4 zachytenom v analyzačnom programe WireShark, môžeme vidieť presné naplnenie polí, podľa ich výskytu v šablóne. Spodné hexadecimálne hodnoty značia aká hodnota je v danom polí obsiahnutá. Následné parsovanie hodnôt prebiehalo práve na základe týchto hexadecimálnych údajov a ich výskytu(indexu) v UDP paketových dátach.

2.2 Programovanie NetFlow v9 parseru v jazyku Python

Programovanie prebiehalo vo verzií programovacieho jazyku Python 3.7, program bol otestovaný aj na verzií 3.6 a 3.8 . Na programovanie som použil ako návod práve zachytené exportné pakety a výpis ich dát ako hexdump súbory v programe Wireshark. Ukážky a kódy nie sú kompletné, kompletný program je v prílohe.

2.2.1 Čiastočné ukážky a popis obsiahnutých tried, premenných a atribútov programu

Výpis 2.1: Ukážka slovníkovej definície polí a definícia triedy NetFlowPacket a jej atribútov

```
FIELD_DESCRIPTION = {
    1: {'field_type': 'BYTES', 'length': 4, 'title':
  'Octets'},
    2: {'field_type': 'PKTS', 'length': 4, 'title':
  'Packets'},
    3: {'field_type': 'FLOWS', 'length': 4},
    4: {'field_type': 'PROTOCOL', 'length': 1, 'title':
  'Protocol'},
    5: {'field_type': 'IP_TOS', 'length': 1, 'title':
  'IP ToS'},
    6: {'field_type': 'TCP_FLAGS', 'length': 1, 'title':
  'TCP Flags'},
    7: {'field_type': 'L4_SRC_PORT', 'length': 2, 'title':
  'SrcPort'},
    8: {'field_type': 'IPV4_SRC_ADDR', 'length': 4, 'title':
  'SrcAddr','func': number_to_ip},
    9: {'field_type': 'SRC_MASK', 'length': 1, 'title':
  'SrcMask'},
    10: {'field_type': 'INPUT_SNMP', 'length': 2, 'title':
  'InputInt'},

class NetflowPacket:
    def __init__(self, header, flowset):
        self.header = header
        self.data = flowset
```

V prvom výpise 2.1 kódu môžeme vidieť definovanie niektorých jednotlivých polí podľa štruktúry danej spoločnosťou Cisco, celá definícia polí a ich veľkosti sú ako príloha na konci tejto práce. Následné metódou *class NetflowPacket* incializujem objekty a ich atribúty.

Výpis 2.2: Pokračovanie v definícií tried hlavného súboru netflow_packet.py

```
def create(cls, data, templates):
    header = NetflowHeader.create(data)
    flowset = NetflowFlowset.create(skip_header(data),
    templates)

    return cls(header, flowset)

def __repr__(self):
    repr = ''
    repr += str(self.header)
    for k, item in enumerate(self.data):
        if item.is_flowset_with_only_template():
            repr += 'FlowSet {} [id={}] (Data Template)
            : {}\n'.format(k + 1, item.flow_set_id,
            item.flow_set.template_id)
             repr += '\tFlowSet Id: (Data Template) (V9)
             ({})\n'.format(item.flow_set_id)
            repr += '\tFlowSet Length: {}\n'
            .format(item.flowset_length)
            repr += '\t' + str(item.flow_set).replace
            ('\n', '\n\t')
            repr += '\n'
        else:
            repr += 'FlowSet {} [id={}] ({} flows)\n'
            .format(k + 1, item.flow_set_id,
            len(item.flow_set))
```

V ukážke druhého výpisu 2.2 sa najprv zavolajú dve triedy a vytvoria sa im atribúty, následne sa tieto atribúty vracajú v objektoch. Následne pomocou *repr* vypisujem či flowset obsahuje šablónu alebo sa jedná len toky(flowy) s identifikátorom nejakej šablóny.

Výpis 2.3: Definícia triedy NetflowHeader hlavného súboru

```
class NetflowHeader:
 def __init__(self,
              version: int,
              count: int,
              system_uptime: float,
              unix_seconds: datetime,
              package_sequence: int,
              source_id: int
              ):
     self.version = version
     self.count = count
     self.system_uptime = system_uptime
     self.unix_seconds = unix_seconds
     self.package_sequence = package_sequence
     self.source_id = source_id

 @classmethod
 def create(cls, header_data: str):
     version = int(header_data[0:4], 16)
     count = int(header_data[4:8], 16)
     system_uptime = int(header_data[8:16], 16)/1000
     unix_seconds = datetime.fromtimestamp(int
     (header_data[16:24], 16))
     package_sequence = int(header_data[24:32], 16)
     source_id = int(header_data[32:40], 16)

     return cls(version, count, system_uptime,
     unix_seconds, package_sequence, source_id)

     def __repr__(self):
     repr = ''
     repr += 'Version: {}\n'.format(self.version)
     repr += 'Count: {}\n'.format(self.count)
     repr += 'SysUptime: {}\n'.format(self.system_uptime)
     repr += 'Timestamp: {}\n'.format(self.unix_seconds)
     repr += 'FlowSequence: {}\n'
     .format(self.package_sequence)
     repr += 'SourceId: {}\n'.format(self.source_id)
```

Vo výpise 2.3 je definícia jednotlivých atribútov triedy NetflowHeader, ich naplnenie podľa hexdump a následný výpis pomocou *repr*. Jednotlivé polia sa plnia podľa počtu bajtov a sú zväčša plnené hodnotami typu integer.

Výpis 2.4: Definícia triedy NetflowTemplate hlavného súboru

```
class NetflowTemplate:
    def __init__(self, flow_set_id: int, length: int,
    template_id: int, field_count: int, fields: []):
        self.flow_set_id = flow_set_id
        self.length = length
        self.template_id = template_id
        self.field_count = field_count
        self.fields = fields

    @classmethod
    def create(cls, data):
        flow_set_id = int(data[0:4], 16)
        length = int(data[4:8], 16)
        template_id = int(data[8:12], 16)
        field_count = int(data[12:16], 16)
        fields = []

        buffer = data[16:(field_count * 4 * 2 + 16)]
        while len(buffer) > 0:
            field_type = int(buffer[0:4], 16)
            field_length = int(buffer[4:8], 16)

            i = None
            if field_type in FIELD_DESCRIPTION:
                i = FIELD_DESCRIPTION[field_type]

            fields.append({
                'value': field_type,
                'field_type': i['field_type'],
                'length': field_length,
                'title': i['title'] if 'title'
                in i else i['field_type'],
                'func': i['func'] if 'func'
                in i else lambda x: x,
```

Výpis 2.5 ukazuje definovanie triedy NetflowTemplate, priradenie hodnôt dát z jednotlivých polí. Potom definuje buffer a posunie sa o ďalšie dve polia, následne sa použije field_type, ktorý pridá atribúty poli k fields pomocou append na koniec.

Výpis 2.5: Definícia triedy NetflowFlowset hlavného súboru

```
class NetflowFlowset:
    def __init__(self, flow_set, flow_set_id: int,
    flowset_length: int):
        self.flow_set = flow_set
        self.flow_set_id = flow_set_id
        self.flowset_length = flowset_length

    @classmethod
    def create(cls, data: str, templates:
    List[NetflowTemplate]):
        flowsets = []
        buffer = data
        while len(buffer) > 0:
            flowset_length = int(buffer[4:8], 16) * 2
            content = buffer[0:flowset_length]

            if NetflowTemplate.is_template(buffer):
                template = NetflowTemplate.create(content)
                flowsets.append(cls(template, int(buffer[0:4],
                16),
                int(flowset_length/2)))
                templates.append(template)
            else:
                flowsets.append(cls(NetflowDataFlow.create
                (content, templates),
                int(buffer[0:4], 16), int(flowset_length/2)))

            buffer = buffer[flowset_length:]
        return flowsets
```

Tak ako v predošlom kroku som definoval NetflowTemplate triedu s danými objektami podobne postupujem aj pri triede NetflowFlowset. Následne sa bufferom posunieme na dáta, kde skontrolujem dĺžku flow setu a vytvorím content s daným obsahom. Potom definujem šablónu alebo dáta priradím flowset podľa podmienky if, kde obsahuje buď šablónu alebo dáta.

Výpis 2.6: Definícia triedy NetflowDataFlow hlavného súboru

```
class NetflowDataFlow:
    def __init__(self, data: list, template:NetflowTemplate):
        self.data = data
        self.template = template

    @classmethod
    def create(cls, data: str, templates:List[NetflowTemplate])
    -> List['NetflowDataFlow']:
        flowset_id = int(data[0:4], 16)

        err = True
        for template in templates:
            if template.template_id == flowset_id:
                err = False
                break

        if err:
            raise ValueError('template {} not found'
            .format(flowset_id))

        data_len = len(data) / 2
        if (data_len - 4) % template.fields_length != 0:
            logging.warning('problem with length of flows')
          #  return []

        data = data[8:]
        output = []
        while len(data) > 0:
            tmp = []
            flow = data[0:(template.fields_length * 2)]
            for field in template.fields:
                tmp.append(int(flow[:field['length']*2], 16))
                flow = flow[field['length']*2:]
```

Posledná trieda v hlavnom súbore je trieda NetflowDataFlow, tak ako pri predošlých, priradil som jej objekty a atribúty. Zoznam dostupných šablón je NetflowTemplate a výstupom zoznam tokov NetflowDataFlow.

Neskôr som do flowset_id zapísal dáta, ktoré som porovnal s template_id, ak toto nesedí, tak flowset nemá šablónu s daným identifikátorom. Následne sa plnia dáta a priradzuje sa output na koniec NetflowDataFlow. Výpis potom pokračuje pomocou repr, kde sa polia vypisujú podľa template fields.

Výpis 2.7: Zachytenie UDP eth paketu

```
def extract_udp_payload(eth_packet: Ether):
    return binascii.hexlify(bytes(eth_packet[UDP].
    payload)).decode()

def is_netflow_packet(eth_packet: Ether) -> bool:
    stats['total_packets'] += 1
    if 'type' not in eth_packet.fields:
        logging.debug('packet ignored: type
        not found in eth packet')
        return False

    if eth_packet.type != 0x0800:
        logging.debug('packet ignored: is not IPv4 packet')
        return False

    if eth_packet[IP].proto != 0x11:
        logging.debug('packet ignored: is not UDP packet')
        return False

    bin_data = extract_udp_payload(eth_packet)

    # Cisco NetFlow/IPFIX v9
    if not bin_data.startswith('0009'):
        logging.debug('packet ignored: is not NetFlow v9')
        return False

    stats['total_netflow_packets'] += 1
    return True
```

Vo výpise 2.7 extrahujem príchodzie dáta, paket následne pomocou podmienok podrobujem kontrole a potom v packet.type jednou takou podmienkou zisťujem akú ma hodnotu, rovnakú vec neskôr robím pre zistenie či sa jedná o UDP paket a následne aj či sa jedná o Netflow v9 alebo IPFIX paket.

2.2.2 Testy programovej funkčnosti a atribútov

Výpis 2.8: Testovanie naplnenia headeru pomocou zachytených pcap súborov

```
def test_create_header_from_file_01_data_and_template
    (next_packet):g = next_packet('01_data_and_template.pcap')
    first_packet = next(g)

    header1 = NetflowHeader.create(extract_udp_payload
    (first_packet))
    assert header1.version == 9
    assert header1.count == 2
    assert header1.system_uptime == 209844.824
    assert header1.unix_seconds == datetime(2020, 5, 25, 21,
    12, 36)
    assert header1.package_sequence == 7697
    assert header1.source_id == 0

    second_packet = next(g)
    header2 = NetflowHeader.create(extract_udp_payload
    (second_packet))
    assert header2.version == 9
    assert header2.count == 1
    assert header2.system_uptime == 209877.824
    assert header2.unix_seconds == datetime(2020, 5, 25, 21,
    13, 9)
    assert header2.package_sequence == 7698
    assert header2.source_id == 0

def test_parse_template_02_data_and_template_two_flow
    (next_packet):
    g = next_packet('02_data_and_template_two_flow.pcap')
    first_packet = next(g)

    header1 = NetflowHeader.create(extract_udp_payload
    (first_packet))
    assert header1.version == 9
    assert header1.count == 3
    assert header1.system_uptime == 8467.824
```

```
    assert header1.unix_seconds == datetime(2020, 5, 23, 13,
    16, 19)
    assert header1.package_sequence == 271
    assert header1.source_id == 0
```

Pomocou jednoduchého testovacieho programu som testoval správne naplnenie jednotlivých premenných v triede NetflowHeader. Využil som k tomu funkciu *assert*, ktorá funguje na princípe očakávaného vstupu dát a porovnáva ich so zadanými dátami(pass/false). Podobných testov je v programe viacero, aj pre ostatné triedy a aj databázu a využívajú sa k tomu dáta z pcap súborov z odchytávania pomocou Wiresharku na školskom servery a laboratórnom rozhraní. Nižšie môžeme vidieť aj databázový test 2.11, ktorý vytvára a napĺňa polia jednoduchej databázy.

Výpis 2.9: Testovanie naplnenia databázových polí

```
def test_db_create_field(db):
    field = Field.create(db, 't_field', 1, 2)
    db.session.commit()

    assert field == Field.get_by_name(db, 't_field')
    assert field.name == 't_field'
    assert field.value_id == 1
    assert field.length == 2

def test_db_create_value(db):
    value = Value.create(db, '45')
    db.session.commit()

    assert value == Value.get_last(db)
    assert value.value == '45'

def test_db_create_data(db):
    data = Data.create(db, 'field', 1, 2, '45')
    db.session.commit()

    assert data.field.name == 'field'
    assert data.field.value_id == 1
    assert data.field.length == 2
    assert data.value.value == '45'
```

Posledným dôležitým programom je program cli.py, kde prebiehajú celkové príkazy na spustenie parsovania zo súboru(file) alebo zachytávanie na danom adaptéry(sniff), prípadne dokáže vypísať rozhrania v systéme spolu s adresami.

Výpis 2.10: Spúšťací program cli.py

```
@main.command('file')
@click.argument('filename', type=click.Path(), required=True)
@click.option('-v', '--verbose', 'verbose', is_flag=True,
default=False, help='verbose output')
@click.option('-o', '--output-file', 'output_file',
type=str, default=None, help='name of output file')
def file_cmd(filename, verbose, output_file):
    print('filename: {}, verbose: {}, output-file: {}'
    .format(filename, verbose, output_file))
    for (data, metadata) in RawPcapReader(filename):
        process_packet(data, verbose, output_file)
    print(json.dumps(stats, indent=4))

@main.command('sniff')
@click.argument('interface', required=True)
@click.option('-v', '--verbose', 'verbose', is_flag=True,
default=False, help='verbose output')
@click.option('-o', '--output-file', 'output_file', type=str,
default=None, help='name of output file')
def sniff_cmd(interface, verbose, output_file):
    print('interface: {}'.format(interface))
    sniff(filter='udp', iface=interface,
    prn=process_packet_callback(verbose, output_file))
    print(json.dumps(stats, indent=4))

@main.command('interfaces')
def interfaces():
    from scapy.arch.windows import show_interfaces
    show_interfaces()

if __name__ == '__main__':
    setup_logging(logging.DEBUG)
    main()
```

2.3 Zachytenie reálnych dát vytvoreným parserom

Pomocou príkazu "python cli.py interfaces", som si vyhľadal rozhranie s IP adresou 10.0.0.20, kde exportér(laboratórny smerovač Cisco 1812) zasiela exportované NetFlow v9 pakety, obr.2.5.

```
C:\Users\xbosel00\Desktop\Server\netflow9_xbosel00>python cli.py interfaces
INDEX   IFACE                                       IP              MAC
14      Gigabitové síťové připojení Intel(R) 82574L #2  10.0.0.20       Vmware:8b:32:78
17      Npcap Loopback Adapter                      127.0.0.1       00:00:00:00:00:00
12      Gigabitové síťové připojení Intel(R) 82574L     147.229.148.213 Microsof:94:16:04
-1      [Unknown] NdisWan Adapter                   None            ff:ff:ff:ff:ff:ff
-2      [Unknown] NdisWan Adapter                   None            ff:ff:ff:ff:ff:ff
-3      [Unknown] NdisWan Adapter                   None            ff:ff:ff:ff:ff:ff
16      Microsoft ISATAP Adapter #3                 None
18      Microsoft ISATAP Adapter                    None
15      Microsoft ISATAP Adapter #2                 None
```

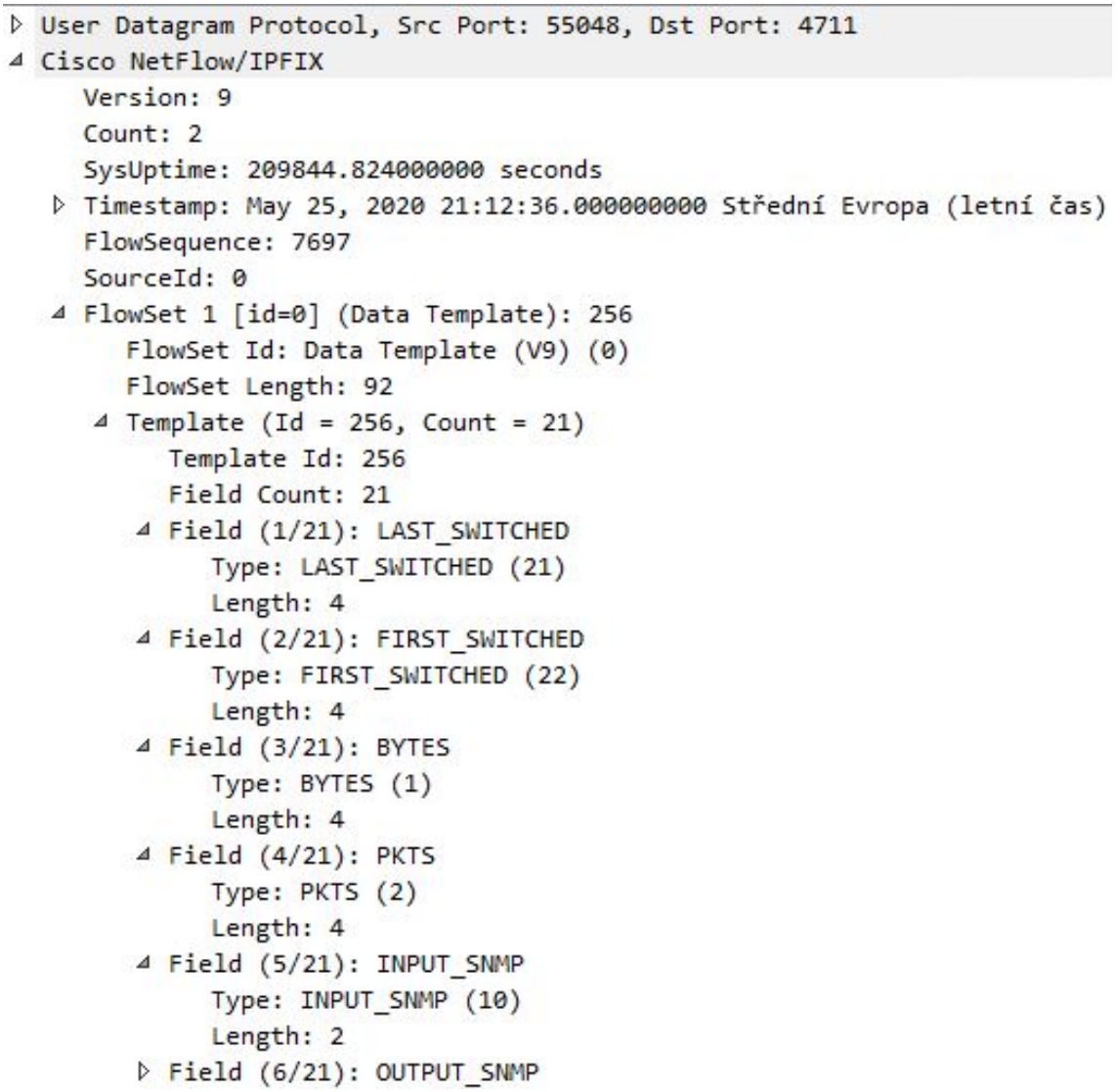
Obr. 2.5: Sieťové rozhrania na školskom servery

Následne sme použili príkaz "python cli.py file data/01_data_and_template.pcap -v", ktorým som rozparsoval a zobrazil dáta z NetFlow v9 paketov v pcap súore, ktoré som predtým zachytili programom Wireshark. Takto som overil správnu funkčnosť parsovania a zhodu dát s dátami vo Wiresharku, na uloženom exportovanom pakete. Ukážka pár zachytených polí obr.2.6

```
C:\Users\xbosel00\Desktop\Server\netflow9_xbosel00>python cli.py file data/01_data_and_template.pcap -v
filename: data/01_data_and_template.pcap, verbose: True, output-file: None
Version: 9
Count: 2
SysUptime: 209844.824
Timestamp: 2020-05-25 21:12:36
        CurrentSecs: 1590433956
FlowSequence: 7697
SourceId: 0
FlowSet 1 [id=0] (Data Template): 256
        FlowSet Id: (Data Template) (V9) (0)
        FlowSet Length: 92
        Template (Id = 256, Count = 21)
                Template Id: 256
                Field Count: 21
                Field (1/21): LAST_SWITCHED
                        Type: LAST_SWITCHED (21)
                        Length: 4
                Field (2/21): FIRST_SWITCHED
                        Type: FIRST_SWITCHED (22)
                        Length: 4
                Field (3/21): BYTES
                        Type: BYTES (1)
                        Length: 4
                Field (4/21): PKTS
                        Type: PKTS (2)
                        Length: 4
                Field (5/21): INPUT_SNMP
                        Type: INPUT_SNMP (10)
                        Length: 2
                Field (6/21): OUTPUT_SNMP
                        Type: OUTPUT_SNMP (14)
                        Length: 2
                Field (7/21): IPV4_SRC_ADDR
                        Type: IPV4_SRC_ADDR (8)
                        Length: 4
                Field (8/21): IPV4_DST_ADDR
                        Type: IPV4_DST_ADDR (12)
                        Length: 4
                Field (9/21): PROTOCOL
                        Type: PROTOCOL (4)
                        Length: 1
```

Obr. 2.6: Zachytená šablóna a pár prvých polí ktoré definuje

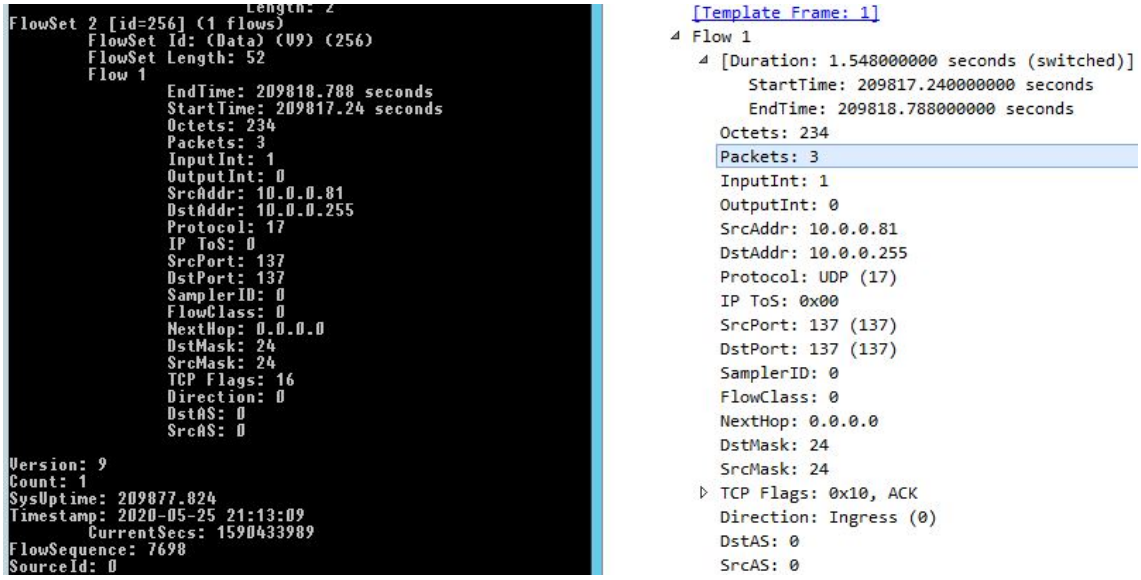
Na nasledujúcom obrázku 2.7 môžeme vidieť, že polia vo Wiresharku, zodpovedajú presne parsovaným dátam vo výpise príkazového riadku na servery. Teda je parsovanie nastavené a naprogramované správne.



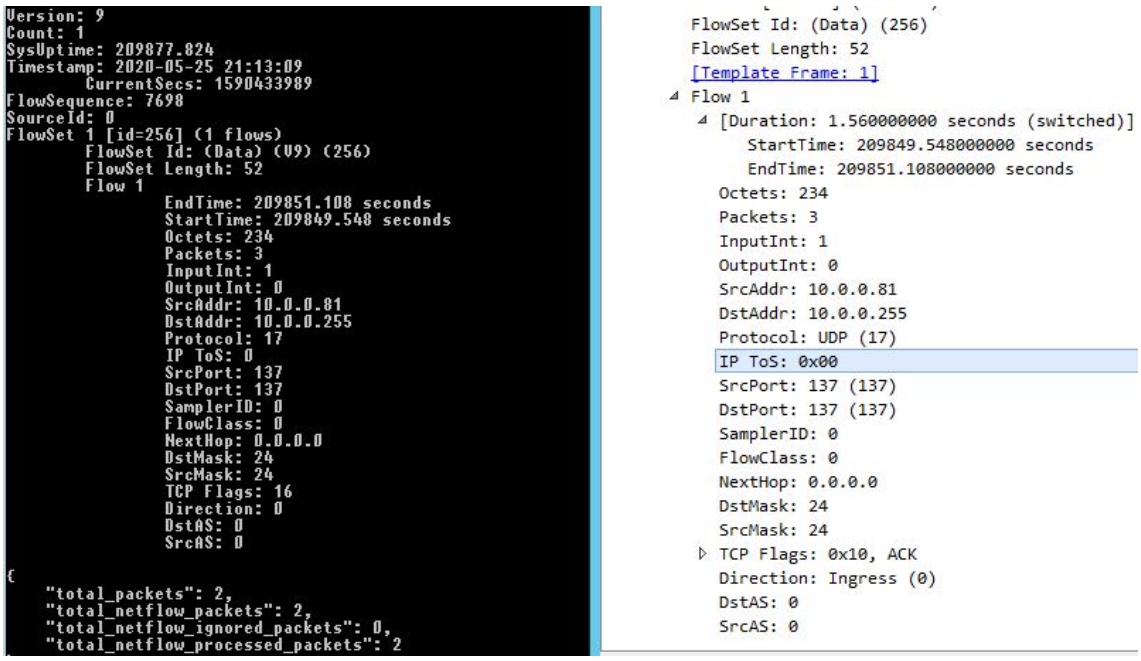
Obr. 2.7: Šablóna vo Wiresharku s definovanými poliami

Ďalej som sa zameral na potvrdenie správnosti údajov a dát, ktoré naplnia jednotlivé polia podľa šablóny. V dátach, ktoré vypisujem sú dva pakety(teda de facto dva flowy), ktoré sa držia jednej šablóny - 256, obr.2.8 a obr.2.9.

Pomocou príkazu *python cli.py sniff "Gigabitové síťové připojení Intel(R) 82574L #2-verbose*, som zachytil aj dáta na porte, prípona verbose znamená detailný výpis, obr.2.10 a 2.11.

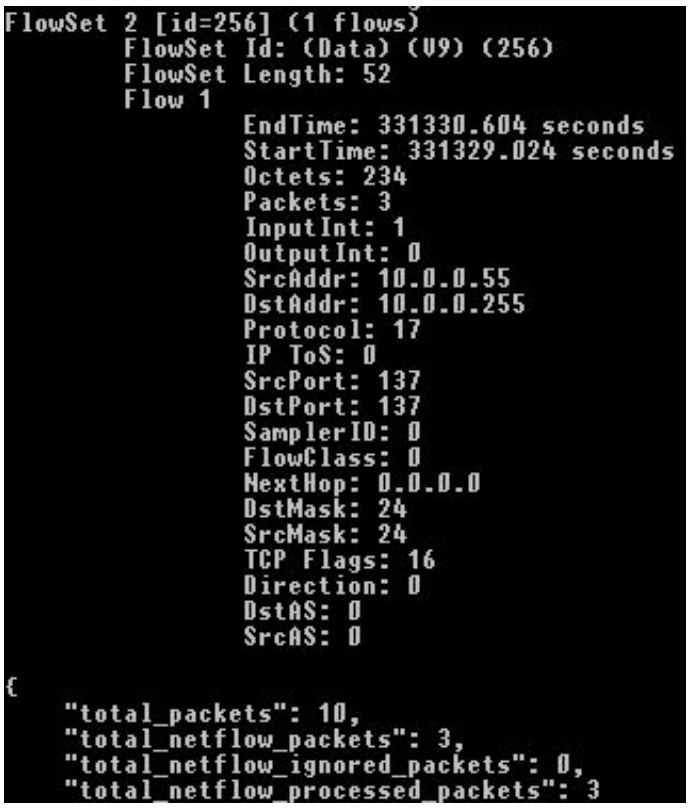


Obr. 2.8: Dáta na porovnanie s Wiresharkom - paket 1



Obr. 2.9: Dáta na porovnanie s Wiresharkom - paket 2

Obr. 2.10: Zachytená šablóna definujúca polia na porte



Obr. 2.11: Dáta podľa prijatej šablóny na porte

2.3.1 Ukladanie dát do jednoduchej databázy

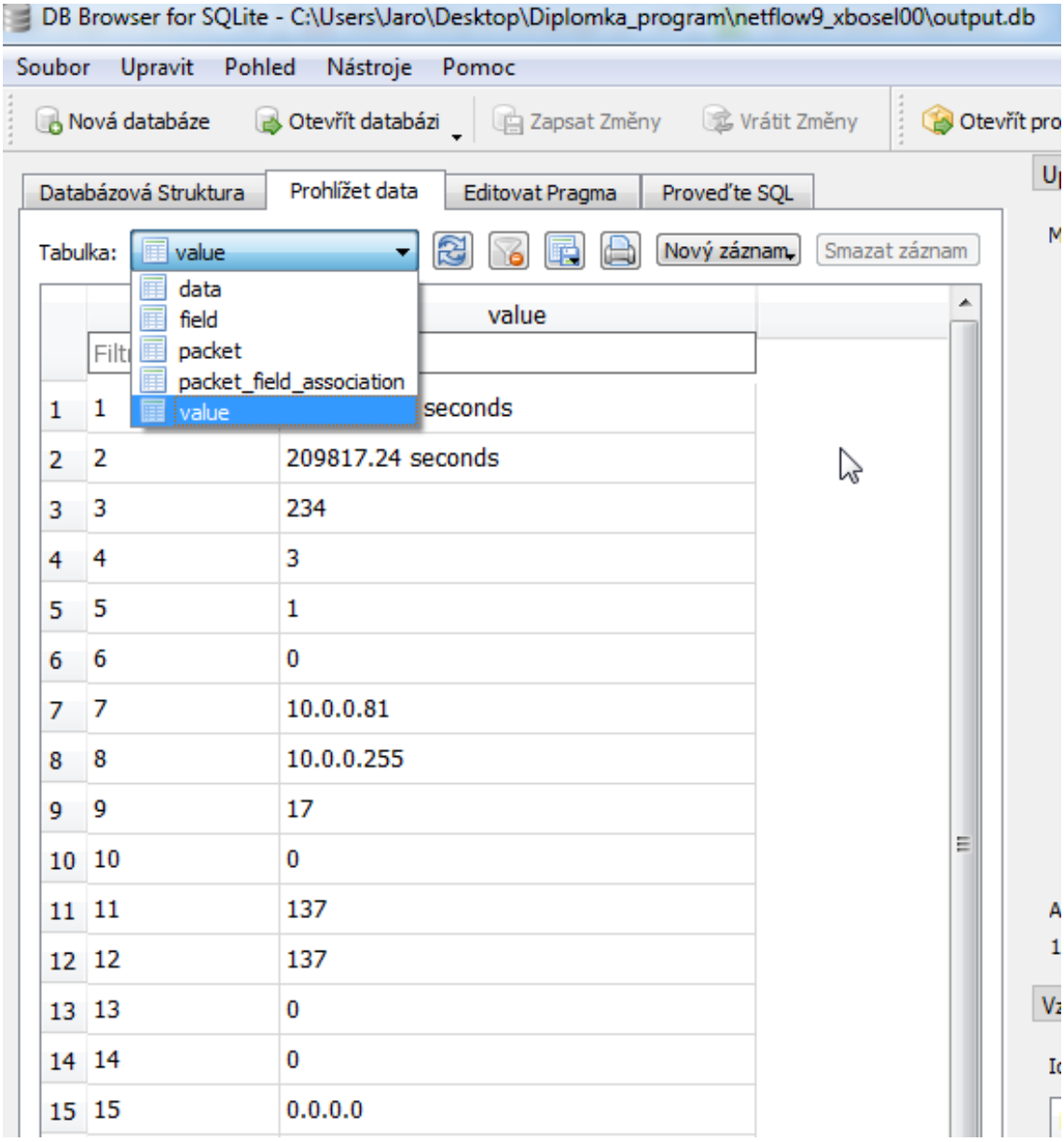
V poslednom kroku som naprogramoval jednoduchú databázu, ktorá v podstate len napĺňa pár hodnôt z exportovaných dát. Do databázy napĺňam základné dáta o poliach, samotné polia, hodnoty ktoré v nich sú, paket a pričlenené polia k paketu. Na prezeranie databázy som použil DB Browser for SQLite na Windows.

Výpis 2.11: Ukážka programovania databázy

```
class Data(Base):
    __tablename__ = 'data'
    id = Column(Integer, primary_key=True)
    field_id = Column(Integer, ForeignKey('field.id'),
    nullable=False)
    value_id = Column(Integer, ForeignKey('value.id'),
    nullable=False)

    field = relationship('Field')
    value = relationship('Value')

    def __init__(self, field, value):
        self.field = field
        self.value = value

    @classmethod
    def create(cls, db, field_name, field_value_id,
    field_length,value_value):
        field = Field.create(db, field_name, field_value_id,
        field_length)
        value = Value.create(db, value_value)
        result = cls(field, value)
        db.session.add(result)
        return result
```

Pomocou príkazu *python cli.py sniff "Gigabitové síťové připojení Intel(R) 82574L #2o output.db*, je možné zachytávať dáta na porte a potom ich zapísať do databázy - output(môže mať ľubovoľný názov). Ukážka takého zachytávania na obr.2.13.

Obr. 2.12: Zachytávanie dát na porte a ich čiastočné ukladanie do databázy



Obr. 2.13: SQLite DB Browser program a v ňom naplnené hodnoty zo zachytených NetFlow v9 paketov

Záver

V práci som sa zoznámil s internetovými protokolmi ktoré definujú informácie o sieťovom toku, opisujú jeho štruktúru, veľkosť a vlastnosti. V teoretickej časti práce som obecne opísal sieťovú prevádzku, dátový tok a typy prenosov. Ďalej som podrobne opísal proprietárny protokol spoločnosti Cisco - NetFlow v9, hlavne som sa zameral na jeho deviatu verziu, ktorá definuje šablóny, zároveň s ním som opísal aj protokol IPFIX.

Opísal som formát správ, jednotlivé polia a informácie, ktoré do nich prichádzajú. Tieto informácie som použil pri programovaní NetFlow v9 parseru ktorý je využitý pri zachytávaní dát a parsovaní informácií. V teoretickej časti som sa ešte zameral na protokol IPFIX, ktorý je založený na protokole NetFlow v9, avšak je širko štandardizovaný. Opísal som rozdiely v informáciach ktoré IPFIX nesie, ktoré polia v správe má oproti NetFlow naviac a hlavné odlišnosti od protokolu NetFlow. V rámci teórie som ešte v skrátennej verzií spomenul ďalšie tri informačné protokoly, ktoré prenášajú záznamy o toku - sFlow, NetStream a OpenFlow. Urobil som prehľad o týchto protokoloch a opísal ich základné funkcionality. Niektoré z nich sú veľmi podobné protokolu NetFlow v9 alebo IPFIX, avšak definujú si svoje vlastné proprierátne polia a štruktúry.

V praktickej časti práce, som najprv nakonfiguroval fyzický smerovač Cisco 1812, ktorý je umiestnený v Laboratóriu Transporných síti na Technickej 12 - FEKT, SC.5.35. Konfiguroval som ho podľa konfiguračných súborov spoločnosti Cisco, rovnako som pomocou ich príkazov overil informácie o generovaní NetFlow v9 paketov, šablón(template) a asociovaných tokoch(flowoch) s danými šablónami. Následne som zachytil exportné pakety pomocou programu WireShark, pakety som dekódoval ako CFLOW tok a analyzoval ich po poliach a hexadátach. Po tejto analýze som začal na základe polí a hexdump hodnôt jednotlivých dát vo WireSharku, programovať a implementovať ucelený program na parsovanie dát, informácií a šablón z NetFlow v9 paketov. Program som otestoval na zachytených pcap súboroch, obsahujúce NetFlow v9 pakety s požadovanými dátami a informáciami.

Samotný program obsahuje jednoduché testy na základnú funkčnosť(napĺňanie tried, testovanie databázy). Okrem pcap súborov bol program otestovaný aj pri odchytávaní NetFlow paketov na porte 4711 na školskom servery s použitím laboratórnym sieťovým adaptérom.

V práci sú definované len hlavné a niektoré ukážkové časti kódu, celý kód je priložený ako príloha k prácí, aj so súborom Readme_file, kde sú definované príkazy do príkazového riadku na spustenie kódu. V súbore je celá štruktúra kódu a zachytené testovacie súbory s dátami(pcap), ako aj testy programu.

Z programu sú výstupné dáta, ktoré sa môžu zapísať do jednoduchej databázy ktorú som si pripravil. Na záver som si databázu zobrazil dostupným bezplatným SQL prehliadačom pre Windows a tým potvrdil funkčnosť kolektoru a zachytené NetFlow v9 dáta a šablóny(fungujúce zachytávanie na porte a parsovanie, parsovanie zo súboru). Po programátorskej stránke nie je program úplne dokonalý, avšak bola overená jeho funkčnosť a stabilita v prostredí školského laboratória, a na vývojovom servery.

Literatúra

[1] Techpedia *Network Traffic* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `http://bit.ly/2Kb5p57`.

[2] Ghassan A. QasMarrogy, Dr. Emmanuel S. QasMarrogy, Aous Y. Ali *Performance Analysis of Real and Non-real Time Traffic under MANET Routing Protocols* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `http://bit.ly/2DeuWXg`.

[3] All About Circuits *Data Flow* Data Flow - chapter 14 - Digital Communication [online]. [cit. 25. 11. 2019]. Dostupné z URL: `http://bit.ly/2YvPAvR`.

[4] KOZIEROK, M., Charles *TCP/IP Guide*. 1. vyd. No Starch Press, San Francisco, 2005. 1618 s. ISBN-13: 978-1-59327-047-6

[5] LUCAS, W., Michael *NETWORK FLOW ANALYSIS*. 2. vyd. No Starch Press, San Francisco, 2010. 228 s. ISBN 1-59327-203-0

[6] CISCO SYSTEMS, INC. CISCO *Introduction to Cisco IOS NetFlow - A Technical Overview* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://bit.ly/1NaU6lT`.

[7] CLAISE, B., Ed., Cisco Systems, Inc. *Cisco Systems NetFlow Services Export Version 9* RFC Cisco Systems NetFlow Services Export Version 9 [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://tools.ietf.org/html/rfc3954`.

[8] PHAAL, P., PANCHEN,S., MCKEE, N., InMon Corp. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://tools.ietf.org/html/rfc3176`.

[9] CISCO SYSTEMS, INC. CISCO *NetFlow Configuration Guide, Cisco IOS Release 15M&T* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `http://bit.ly/2rNnqk8`.

[10] Chapple, Mike, Ph.D *NetFlow Security Monitoring for Dummies* [kniha]. Hoboken, NJ: John Wiley & Sons, Inc., 2007, [cit. 11. 11. 2019]. ISBN 978-1-118-33772-1.

[11] Garland Technology Inc. *Network Test Access Point (TAP) and Port Mirroring (SPAN)* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://www.garlandtechnology.com/tap-vs-span`.

[12] CISCO SYSTEMS, INC. CISCO, Caligare web *NETFLOW PACKET VERSION 9 (V9)* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://netflow.caligare.com/netflow_v9.htm`.

[13] CISCO SYSTEMS, INC. CISCO *Cisco IOS NetFlow Version 9 Flow-Record Format* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://bit.ly/2A4Fslz`.

[14] CLAISE, B., Ed., TRAMMELL, B., Ed., AITKEN, P. Cisco Systems, Inc., ETH Zurich *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* RFC Cisco Systems NetFlow Services Export Version 9 [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://tools.ietf.org/html/rfc7011`.

[15] CLAISE, B., TRAMMELL, B. *Applying IPFIX to Network Measurement and Management* IETF IPFIX Presentation [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://www.ietf.org/slides/slides-edu-ipfix-00.pdf`.

[16] GRAHAM, M. *Feature Comparison between NetFlow v5 NetFlow v9 and IPFIX summarised* ResearchGate Table [online]. [cit. 25. 11. 2019]. Dostupné z URL: `http://bit.ly/2ZbL4Tx`.

[17] sFlow konzorcium *sFlow Overview & Specification* About sFlow [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://sflow.org/about/index.php`.

[18] Huawei Technologies Co. Ltd. *Technical White Paper for NetStream* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `http://bit.ly/34IXO5l`.

[19] Dijiang Huang, Huijun Wu, *Openflow Protocol* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://www.sciencedirect.com/topics/computer-science/openflow-protocol`.

[20] Sdx Central *What is OpenFlow? Definition and How it Relates to SDN* SDN [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://www.sdxcentral.com/networking/sdn/definitions/what-is-openflow/`.

[21] CISCO SYSTEMS, INC. CISCO *NetFlow Configuration Guide* [online]. [cit. 25. 11. 2019]. Dostupné z URL: `https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/netflow/configuration/xe-16/nf-xe-16-book/cfg-nflow-data-expt-xe.html`.

[22] PILGRIM, Mark *Ponořme se do Python(u)3: Dive into Python 3*. CZ.NIC, c2010.CZ.NIC, Praha, 2010. 228 s. ISBN 978-80-904248-2-1

Zoznam symbolov, veličín a skratiek

ATM	Asynchronný prenosový mód – Asynchronous Transfer Mode
HTTPS	Hypertext Transfer Protocol Secure – Hypertextový bezpečný prenosový protokol
TCP	Transmission Control Protokol – Protokol Riadenia prenosu
UDP	User Datagram Protocol – Datagramový protokol
WAN	Wide Area Network — Rozľahlá oblastná sieť
QoS	Quality of Service — Kvalita služby
LAN	Local Area Network — Lokálna sieť
DOS	Denial of Service — Odmietnu-tie Služby
SCTP	Stream Control Transmission Protocol – Kontrolný protokol prenosu toku
SDN	Software Defined Network – Softvérovo definovaná sieť
SPAN	Switchport analyzer – Analýzer na porte prepínača
TAP	Test Access Point – Prísupový bod na zbieranie toku
IPFIX	Internet Protocol Flow Export – Internetový protokol na exportovanie toku
SSH	Secure Shell – Zabezpečený terminál

Zoznam príloh

A Ukážka a popis polí v NetFlow v9

Typ poľa	Hodnota	Bajty	Popis
IN_BYTES	1	N(Predvolené sú 4)	Príchodzí počet bajtov v IP toku
IN_PKTS	2	N(Predvolené sú 4)	Príchodzí počet paketov v IP toku
FLOWS	3	N(Predvolené sú 4)	Počet agregovaných tokov
PROTOCOL	4	1	IP Protokol
TOS	5	1	Typ služieb bajt vstupného prichádzajúceho rozhrania
TCP_FLAGS	6	1	Kumulácia TCP flagov v danom toku
L4_SRC_PORT	7	2	TCP/UDP zdrojový port
IPV4_SRC_ADDR	8	4	IPv4 zdrojová adresa
SRC_MASK	9	1	Maska zdrojovej adresy
INPUT_SNMP	10	N (predvolené sú 2)	Vstupne rozhranie indexu
L4_DST_PORT	11	2	TCP/UDP cieľový port
IPV4_DST_ADDR	12	4	IPv4 cieľová adresa
DST_MASK	13	1	Maska cieľovej adresy
OUTPUT_SNMP	14	N (predvolené sú 2)	Výstupné rozhranie indexu
IPV4_NEXT_HOP	15	4	IPv4 adresa ďalšieho smerovača
SRC_AS	16	N (predvolené sú 2)	Číslo zdrojového BGP autonómného systému
DST_AS	17	N (predvolené sú 2)	Číslo cieľového BGP autonómného systému
BGP_IPV4_NEXT_HOP	18	4	IPv4 adresa ďalšieho routra v BGP doméne
MUL_DST_PKTS	19	N(Predvolené sú 4)	Počet IP multicast odchádzajúcich paketov
MUL_DST_BYTES	20	N(Predvolené sú 4)	Počet IP muticast odchádzajúcich bajtov

LAST_SWITCHED	21	4	Uptime posledného paketu v toku v ms
FIRST_SWITCHED	22	4	Uptime prvého paketu v toku v ms
OUT_BYTES	23	N(Predvolené sú 4)	Počet odchádzajúcich bajtov
OUT_PKTS	24	N(Predvolené sú 4)	Počet odchádzajúcich paketov
MIN_PKT_LNGTH	25	2	Minimálna dĺžka prichádzajúcich paketov
MAX_PKT_LNGTH	26	2	Maximálna dĺžka prichádzajúcich paketov
IPV6_SRC_ADDR	27	16	IPv6 zdrojová adresa
IPV6_DST_ADDR	28	16	IPv6 cieľová adresa
IPV6_SRC_MASK	29	1	Maska zdrojovej adresy IPv6
IPV6_DST_MASK	30	1	Maska cieľovej adresy IPv6
IPV6_FLOW_LABEL	31	3	Značka IPv6 toku
ICMP_TYPE	32	2	Typ ICMP paketu
MUL_IGMP_TYPE	33	1	Typ IGMP paketu
SAMPLING_INTERVAL	34	4	Interval množstva vzorkovaných paketov
SAMPLING_ALGORITHM	35	1	Čas pre aktívne toky
FLOW_ACTIVE_TIMEOUT	36	2	IPv4 adresa ďalšieho smerovača
FLOW_INACTIVE_TIMEOUT	37	2	Čas pre neaktívne toky
ENGINE_TYPE	38	1	Typ zariadenia prepínajúceho toky
ENGINE_ID	39	1	ID zariadenia
TOTAL_BYTES_EXP	40	N(Predvolené sú 4)	Počet exportovaných bajtov
TOTAL_PKTS_EXP	41	N(Predvolené sú 4)	Počet exportovaných paketov
TOTAL_FLOWS_EXP	42	N(Predvolené sú 4)	Počet exportovaných tokov
IPV4_SRC_PREFIX	44	4	IPv4 prefix zdrojovej adresy
IPV4_DST_PREFIX	45	4	IPv4 prefix cieľovej adresy
MPLS_TOP_LABEL_TYPE	46	1	Top typ pre MPLS
MPLS_TOP_LABEL_IP_ADDR	47	4	Ekvivalentná trieda odpovedajúca MPLS Top typu
FLOW_SAMPLER_ID	48	1	ID “show flow sampler”

FLOW_SAMPLER_MODE	49	1	Typ algoritmu pre vzorkovanie dát
FLOW_SAMPLER_RANDOM_INTERVAL	50	4	Interval vzorkovania
MIN_TTL	52	1	Minimálne TTL v prichádzajúcich paketoch toku
MAX_TTL	53	1	Maximálne TTL v prichádzajúcich paketoch toku
IPV4_IDENT	54	2	IPv4 identifikátor
DST_TOS	55	1	Typ služby výdajného zariadenia
SRC_MAC	56	6	Zdrojová MAC adresa
DST_MAC	57	6	Cieľová MAC adresa
SRC_VLAN	58	2	Zdrojová VLAN
DST_VLAN	59	2	Cieľová VLAN
IP_PROTOCOL_VERSION	60	1	Verzia IP protokolu - 4 pre IPVv4, 6 pre IPv6
DIRECTION	61	1	Smer toku 0 - vstup, 1 - výstup
IPV6_NEXT_HOP	62	16	IPv6 adresa nasledujúceho routra
BPG_IPV6_NEXT_HOP	63	16	IPv6 adresa nasledujúceho routra v BGP doméne
IPV6_OPTION_HEADERS	64	4	Identifikuje IPv6 hlavičku v toku
MPLS_LABEL_1	70	3	MPLS na pozícii 1 v zásobníku
MPLS_LABEL_2	71	3	MPLS na pozícii 2 v zásobníku
MPLS_LABEL_3	72	3	MPLS na pozícii 3 v zásobníku
MPLS_LABEL_4	73	3	MPLS na pozícii 4 v zásobníku
MPLS_LABEL_5	74	3	MPLS na pozícii 5 v zásobníku
MPLS_LABEL_6	75	3	MPLS na pozícii 6 v zásobníku
MPLS_LABEL_7	76	3	MPLS na pozícii 7 v zásobníku
MPLS_LABEL_8	77	3	MPLS na pozícii 8 v zásobníku
MPLS_LABEL_9	78	3	MPLS na pozícii 9 v zásobníku
MPLS_LABEL_10	79	3	MPLS na pozícii 10 v zásobníku

B Súborový strom programu a samotný program v .zip (ako príloha k dipl.práci)

```
Server
└── netflow9_xbosel00
    ├── data
    │   ├── 01_data_and_template
    │   ├── 02_data_and_template_two_flow
    │   ├── 03_without_template
    │   ├── 04_2_templates_in_flowset
    │   ├── 05_2_templates_with_data
    │   └── netflow
    ├── netflow9
    │   ├── database
    │   │   ├── __init__
    │   │   ├── base
    │   │   ├── data
    │   │   ├── database
    │   │   ├── field
    │   │   ├── packet
    │   │   └── value
    │   ├── protocol
    │   │   ├── __init__
    │   │   └── netflow_packet
    │   ├── util
    │   │   ├── __init__
    │   │   ├── netflow_util
    │   │   ├── package_root_dir
    │   │   └── setup_logging
    │   └── __init__
    ├── tests
    │   ├── test_cases
    │   │   ├── test_db
    │   │   ├── test_netflow_data_flow
    │   │   ├── test_netflow_flowset
    │   │   ├── test_netflow_header
    │   │   ├── test_netflow_packet
    │   │   └── test_netflow_template
    │   └── conftest
    ├── cli.py
    ├── Makefile
    ├── output2.db
    ├── Readme_file.md
    └── requirements
```